Draft Federal Information
Processing Standards Publication 183

1993 December 21

Announcing the Standard for

INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

1.Name of Standard. Integration Definition for Function Modeling (IDEF0).

2.Category of Standard. Software Standard, Modeling Techniques.

3.Explanation. This publication announces the adoption of the Integration Definition Function Modeling (IDEF0) as a Federal Information Processing Standard (FIPS). This standard is based on the Air Force Wright Aeronautical Laboratories Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part II, Volume IV - Function Modeling Manual (IDEF0), June 1981.
This standard describes the IDEF0 modeling language (semantics and syntax), and associated rules and techniques, for developing structured graphical representations of a system or enterprise. Use of this standard permits the construction of models comprising system functions (activities, actions, processes, operations), functional relationships, and data (information or objects) that support systems integration.
This standard is the reference authority for use by system or enterprise modelers required to utilize the IDEF0 modeling technique, by implementors in developing tools for implementing this technique, and by other computer professionals in understanding the precise syntactic and semantic rules of the standard.

4.Approving Authority. Secretary of Commerce.

5.Maintenance Agency. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory.

6.Cross Index.
a. ICAM Architecture Part II-Volume IV - Function Modeling Manual (IDEF0), AFWAL-TR-81-4023, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, June 1981.

7.Related Documents.
a.Federal Information Resources Management Regulations Subpart 201.20.303, Standards, and Subpart 201.39.1002, Federal Standards.

b.Integrated Information Support System (IISS), Volume V - Common Data Model Subsystem, Part 4 - Information Modeling Manual - IDEF1 Extended, December 1985.
c.ICAM Architecture Part II, Volume V - Information Modeling Manual (IDEF1), AFWAL-TR-81-4023, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, June 1981.
d.ICAM Configuration Management, Volume II - ICAM Documentation Standards for Systems Development Methodology (SDM), AFWAL-TR-82-4157, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, October 1983.

8.Objectives. The primary objectives of this standard are:

a.To provide a means for completely and consistently modeling the functions (activities, actions, processes, operations) required by a system or enterprise, and the functional relationships and data (information or objects) that support the integration of those functions;

b.To provide a modeling technique which is independent of Computer-Aided Software Engineering (CASE) methods or tools, but which can be used in conjunction with those methods or tools;

c.To provide a modeling technique that has the following characteristics:

-Generic (for analysis of systems of varying purpose, scope and complexity);
-Rigorous and precise (for production of correct, usable models);
-Concise (to facilitate understanding, communication, consensus and validation);
-Conceptual (for representation of functional requirements rather than physical or organizational implementations);
-Flexible (to support several phases of the lifecycle of a project).

9.Applicability.
The use of this standard is strongly recommended for projects that:

a.Require a modeling technique for the analysis, development, re-engineering, integration, or acquisition of information systems;

b.Incorporate a systems or enterprise modeling technique into a business process analysis or software engineering methodology.

The specifications of this standard are applicable when system or enterprise modeling techniques are applied to the following:

a.Projects requiring IDEF0 as the modeling technique;

b.Development of automated software tools implementing the IDEF0 modeling technique.

The specifications of this standard are not applicable to those projects requiring a function modeling technique other than IDEF0.

Nonstandard features of the IDEF0 technique should be used only when the needed operation or function cannot reasonably be implemented with the standard features alone. Although nonstandard features can be very useful, it should be recognized that the use of these or any other nonstandard elements may make the integration of models more difficult and costly.

10.Specifications. This standard adopts the Integration Definition for Function Modeling (IDEF0) as a Federal Information Processing Standard (FIPS).

11.Implementation. The implementation of this standard involves two areas of consideration: acquisition of implementations and interpretation of the standard.

11.1 Acquisition of IDEF0 Implementations. This publication (FIPS 183) is effective June 30, 1994. For Federal acquisitions after this date, projects utilizing the IDEF0 function modeling technique, or software implementing the IDEF0 modeling technique, should conform to FIPS 183. Conformance to this standard should be considered whether the project or software utilizing the IDEF0 modeling technique is acquired as part of an ADP system procurement, acquired by separate procurement, used under an ADP leasing arrangement, or specified for use in contracts for programming services.
A transition period provides time for industry to develop products conforming to this standard. The transition period begins on the effective date and continues for one (1) year thereafter. The provisions of this publication apply to orders placed after the date of this publication; however, utilizing a function modeling technique that does not conform to this standard may be permitted during the transition period.

11.2 Interpretation of this FIPS.NIST provides for the resolution of questions regarding the implementation and applicability of this FIPS. All questions concerning the interpretation of this standard should be addressed to:

Director, Computer Systems Laboratory
ATTN: FIPS IDEF0 Interpretation
National Institute of Standards and Technology
Gaithersburg, MD 20899

12Waivers. Under certain exceptional circumstances, the heads of Federal departments and agencies may approve waivers to Federal Information Processing Standards (FIPS). The head of such agencies may redelegate such authority only to a senior official designated pursuant to section 3506(b) of Title 44, United States Code. Requests for waivers shall be granted only when:

a.Compliance with a standard would adversely affect theaccomplishment of the mission of an operator of a Federal computersystem, or

b.Compliance with a standard would cause a major adverse financial impact on the operator which is not offset by government-wide savings.

Agency heads may approve requests for waivers only by awritten decision which explains the basis upon which the agency head made the required finding(s). A copy of each such

decision, with procurement sensitive or classified portions clearly identified, shall be sent to:

Director, Computer Systems Laboratory
ATTN: FIPS Waiver Decisions
National Institute of Standards and Technology
Gaithersburg, MD 20899

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Government Affairs of the Senate and shall be published promptly in the Federal Register.

When the determination on a waiver request applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment of such notice.

A copy of the waiver request, any supporting documents, the document approving the waiver request and any supporting and accompanying documents, with such deletions as the agency is authorized and decides to make under 5 U.S.C. Sec. 552 (b), shall be part of the procurement documentation and retained by the agency.

13.Where to Obtain Copies. Copies of this publication are for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. When ordering, refer to Federal Information Processing Standards Publication 183 (FIPSPUB 183) and title. Payment may be made by check, money order, or deposit account.

Introduction:

This standard is composed of normative and informative sections. Compliance with the normative sections (Sections 1 through 3) is required. The informative sections (Annexes A through D) provide additional suggestions and guidance. Compliance with the informative sections is not required to comply with the standard, unless such compliance is stipulated by an organization adopting this standard.

This informative introduction discusses the background and approach of IDEF0 (pronounced I-def zero). It provides the reader with an orientation and approach to the normative sections.

Background:

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, the ICAM program developed a series of techniques known as the IDEF (ICAM Definition) techniques which included the following:

1. 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.

2. IDEF1, used to produce an "information model". An information model represents the structure and semantics of information within the modeled system or subject area.

3. 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.

Currently, 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 Standards (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 pub-

lished by the U.S. Air Force in the early 1980s.

The IDEF0 Approach:

IDEF0 (Integration DEFinition language 0) is based on SADT™ (Structured Analysis and Design Technique™), 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.

IDEF0 may be used to model a wide variety of automated and non-automated systems. For new systems, IDEF0 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).

As a function modeling language, IDEF0 has the following characteristics:

1. It is comprehensive and expressive, capable of graphically representing a wide variety of business, manufacturing and other types of enterprise operations to any level of detail.

2. It is a coherent and simple language, providing for rigorous and precise expression, and promoting consistency of usage and interpretation.

3. It enhances communication between systems analysts, developers and users through ease of learning and its emphasis on hierarchical exposition of detail.

4. It is well-tested and proven, through many years of use in Air Force and other government development projects, and by private industry.

5. It can be generated by a variety of computer graphics tools; numerous commercial products specifically support development and analysis of IDEF0 diagrams and models.

In addition to definition of the IDEF0 language, the IDEF0 methodology also prescribes procedures and techniques for developing and interpreting models, including ones for data gathering, diagram construction, review cycles and documentation. Materials related solely to modeling procedures are presented in the informative annexes of this document.

Table of Contents

This standard describes the modeling language (syntax and semantics) which supports the IDEF0 technique for developing structured graphical representations of a system or subject area. Use of this standard permits the construction of IDEF0 models comprising system functions (actions, processes, operations), functional relationships, and the data and objects that support systems analysis and design, enterprise analysis, and business process re-engineering.

This document provides three normative sections, Sections 1, 2 and 3, which define the language that supports IDEF0 modeling. This section, Section 1, provides an overview of the document. Section 2 defines the key terms used in the normative sections. Section 3 defines the syntax and semantics of the language.

In addition to the three normative sections, this document also provides four informative annexes. Annex A discusses the concepts that underlie IDEF0. Annex B provides guidelines for creating, interpreting and gathering data for IDEF0 diagrams. Annex C describes structured team-oriented procedures (including forms) for IDEF0 model review and validation. Annex D defines the key terms used in the annexes.

This standard covers IDEF0 as defined by the U.S. Air Force Integrated Computer-Aided Manufacturing (ICAM) Function Modeling Manual (IDEF0), June 1981, and commonly practiced by many IDEF users since then.

The primary objectives of this standard are:

1. To document and clarify the IDEF0 modeling technique and how to correctly use it;

2. To provide a means for completely and consistently modeling the functions required by a system or subject area, and the data and objects that inter-relate those functions;

3. To provide a modeling language which is independent of Computer-Aided Software Engineering (CASE) methods or tools, but which can be used in conjunction with those methods or tools;

4. To provide a modeling language that has the following characteristics:

    a) Generic (for analysis of systems and subject areas of varying purpose, scope and complexity);
    b) Rigorous and precise (for production of correct, usable models);
    c) Concise (to facilitate understanding, communication, consensus and validation);
    d) Conceptual (for representation of functional requirements independent of physical or organizational implementations);
    e) Flexible (to support several phases of the life cycle of a project).

This section contains definitions that relate to the normative sections of this document. See Annex D for definitions for the informative annexes. A term, if defined, is defined in either Section 2 or Annex D.

2.1 A-0 Diagram: The special case of a one-box IDEF0 context diagram, containing the top-level function being modeled and its inputs, controls, outputs and mechanisms, along with statements of model purpose and viewpoint.

2.2 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). See Arrow Segment, Boundary Arrow, Internal Arrow.

2.3 Arrow Label: A noun or noun phrase associated with an IDEF0 arrow or arrow segment, specifying its meaning.

2.4 Arrow Segment: A line segment that originates or terminates at a box side, a branch (fork or join), or a boundary (unconnected end).

2.5 Boundary Arrow: An arrow with one end (source or use) not connected to any box on a diagram. Contrast with Internal Arrow.

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

2.7 Box Name: The verb or verb phrase placed inside an IDEF0 box to describe the modeled function.

2.8 Box Number: The number (0 to 6) placed inside the lower right corner of an IDEF0 box to uniquely identify the box on a diagram.

2.9 Branch: A junction (fork or join) of two or more arrow segments.

2.10 Bundling/Unbundling: The combining of arrow meanings into a composite meaning (bundling), or the separation of arrow meanings (unbundling), expressed by arrow join and fork syntax.

2.11 C-Number: A chronological creation number that may be used to uniquely identify a diagram and to trace its history; may be used as a Detail Reference Expression to specify a particular version of a diagram.

2.12 Call Arrow: A type of mechanism arrow that enables the sharing of detail between models (linking them together) or within a model.

2.13 Child Box: A box on a child diagram.

2.14 Child Diagram: The diagram that details a parent box.

2.15 Context: The immediate environment in which a function (or set of functions on a diagram) operates.

2.16 Context Diagram: A diagram that presents the context of a model, whose node number is A-n (n greater than or equal to zero). The one-box A-0 diagram is a required context diagram; those with node numbers A-1, A-2, ... are optional context diagrams.

2.17 Control Arrow: The class of arrows that express IDEF0 Control, i.e., conditions required to produce correct output. Data or objects modeled as controls may be transformed by the function, creating output. Control arrows are associated with the top side of an IDEF0 box.

2.18 Decomposition: The partitioning of a modeled function into its component functions.

2.19 Detail Reference Expression (DRE): A reference (e.g., node number, C-number, page number) written beneath the lower right corner of an IDEF0 box to show that it is detailed and to indicate which diagram details it.

2.20 Diagram: A single unit of an IDEF0 model that presents the details of a box.

2.21 Diagram Node Number: That part of a diagram's node reference that corresponds to its parent box node number.

2.22 For Exposition Only (FEO) Diagram: A graphic description used to expose or highlight specific facts about an IDEF0 diagram. Unlike an IDEF0 graphic diagram, a FEO diagram need not comply with IDEF0 rules.

2.23 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.

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

2.25 Function Name: Same as Box Name.

2.26 Glossary: A listing of definitions for key words, phrases and acronyms used in conjunction with an IDEF0 node or model as a whole.

2.27 ICOM Code: The acronym of Input, Control, Output, Mechanism. A code that associates the boundary arrows of a child diagram with the arrows of its parent box; also used for reference purposes.

2.28 IDEF0 Model: A graphic description of a system or subject that is developed for a specific purpose and from a selected viewpoint. A set of one or more IDEF0 diagrams that depict the functions of a system or subject area with graphics, text and glossary.

2.29 Input Arrow: The class of arrows that express IDEF0 Input, i.e., the data or objects that are transformed by the function into output. Input arrows are associated with the left side of an IDEF0 box.

2.30 Interface: A shared boundary across which data or objects are passed; the connection between two or more model components for the purpose of passing data or objects from one to the other.

2.31 Internal Arrow: An input, control or output arrow connected at both ends (source and use) to a box on a diagram. Contrast with Boundary Arrow.

2.32 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.

2.33 Mechanism Arrow: The class of arrows that express IDEF0 Mechanism, i.e., the means used to perform a function; includes the special case of Call Arrow. Mechanism arrows are associated with the bottom side of an IDEF0 box.

2.34 Model Note: A textual comment that is part of an IDEF0 diagram, used to record a fact not otherwise depicted.

2.35 Node: A box from which child boxes originate; a parent box. See Node Index, Node Tree, Node Number, Node Reference, Diagram Node Number.

2.36 Node Index: A listing, often indented, showing nodes in an IDEF0 model in "outline" order. Same meaning and content as Node Tree.

2.37 Node Number: A code assigned to a box to specify its position in the model hierarchy; may be used as a Detail Reference Expression.

2.38 Node Reference: A code assigned to a diagram to identify it and specify its position in the model hierarchy; composed of the model name (abbreviated) and the diagram node number, with optional extensions.

2.39 Node Tree: The graphical representation of the parent-child relationships between the nodes of an IDEF0 model, in the form of a graphical tree. Same meaning and content as Node Index.

2.40 Output Arrow: The class of arrows that express IDEF0 Output, i.e., the data or objects produced by a function. Output arrows are associated with the right side of an IDEF0 box.

2.41 Parent Box: A box that is detailed by a child diagram.

2.42 Parent Diagram: A diagram that contains a parent box.

2.43 Purpose: A brief statement of the reason for a model's existence.

2.44Semantics: The meaning of the syntactic components of a language.

2.45 Squiggle: A small jagged line that may be used to associate a label with a particular arrow segment or to associate a model note with a component of a diagram.

2.46 Syntax: Structural components or features of a language and the rules that define relationships among them.

2.47 Text: An overall textual (non-graphical) comment about an IDEF0 graphic diagram.

2.48 Title: A verb or verb phrase that describes the overall function presented on an IDEF0 diagram; the title of a child diagram corresponds to its parent box name.

2.49 Tunneled Arrow: An arrow (with special notation) that does not follow the normal requirement that each arrow on a diagram must correspond to arrows on related parent and child diagrams.

2.50 Viewpoint: A brief statement of the perspective of the model.

This section discusses the basic elements of the IDEF0 modeling technique, identifies the basic components of syntax (graphical component) and semantics (meaning), specifies the rules that govern the use of the IDEF0 technique, and describes the types of diagrams used. Although the components of syntax and semantics are very highly interrelated, each one is discussed separately without regard for the actual sequence of construction.

A model is a representation of a set of components of a system or subject area. The model is developed for understanding, analysis, improvement or replacement of the system. Systems are composed of interfacing or interdependent parts that work together to perform a useful function. System parts can be any combination of things, including people, information, software, processes, equipment, products, or raw materials. The model describes what a system does, what controls it, what things it works on, what means it uses to perform its functions, and what it produces.

IDEF0 is a modeling technique based on combined graphics and text that are presented in an organized and systematic way to gain understanding, support analysis, provide logic for potential changes, specify requirements, or support systems level design and integration activities. An IDEF0 model is composed of a hierarchical series of diagrams that gradually display increasing levels of detail describing functions and their interfaces within the context of a system. There are three types of diagrams: graphic, text, and glossary. The graphic diagrams define functions and functional relationships via box and arrow syntax and semantics. The text and glossary diagrams provide additional information in support of graphic diagrams.

IDEF0 is an engineering technique for performing and managing needs analysis, benefits analysis, requirements definition, functional analysis, systems design, maintenance, and baselines for continuous improvement. IDEF0 models provide a "blueprint" of functions and their interfaces that must be captured and understood in order to make systems engineering decisions that are logical, affordable, integratable and achievable. The IDEF0 model reflects how system functions interrelate and operate just as the blueprint of a product reflects how the different pieces of a product fit together. When used in a systematic way, IDEF0 provides a systems engineering approach to:

1. Performing systems analysis and design at all levels, for systems composed of people, machines, materials, computers and information of all varieties - the entire enterprise, a system, or a subject area;

2. Producing reference documentation concurrent with development to serve as a basis for integrating new systems or improving existing systems;

3. Communicating among analysts, designers, users, and managers;

4. Allowing coalition team consensus to be achieved by shared understanding;

5. Managing large and complex projects using qualitative measures of progress;

6. Providing a reference architecture for enterprise analysis, information engineering and resource management.

Further discussion of concepts, philosophy, and roles for participants in IDEF0 modeling projects is presented in the informative annexes of this document.

The structural components and features of a language and the rules that define relationships among them are referred to as the language's syntax. The components of the IDEF0 syntax are boxes and arrows, rules, and diagrams. Boxes represent functions, defined as activities, processes or transformations. Arrows represent data or objects related to functions. Rules define how the components are used, and the diagrams provide a format for depicting models both verbally and graphically. The format also provides the basis for model configuration management.

A box provides a description of what happens in a designated function. A typical box is shown in Figure 1. Each box shall have a name and number inside the box boundaries. The name shall be an active verb or verb phrase that describes the function. Each box on the diagram shall contain a box number inside the lower right corner. Box numbers are used to identify the subject box in the associated text.

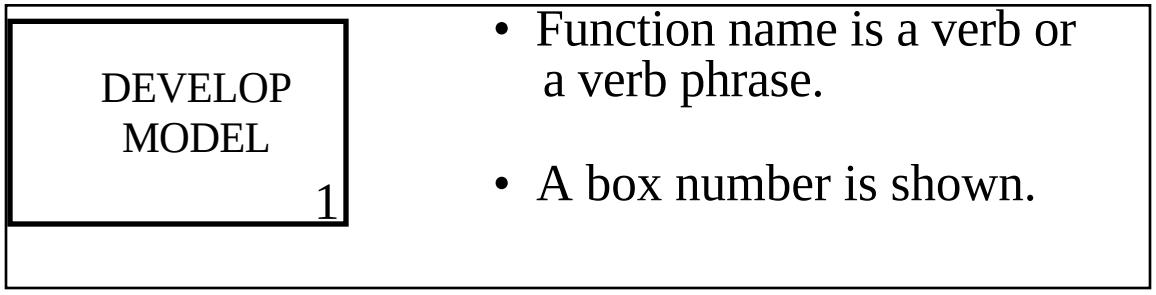


Figure 1. Box Syntax

An arrow is composed of one or more line segments, with a terminal arrowhead at one end. As shown in Figure 2, arrow segments may be straight or curved (with a 90° arc connecting horizontal and vertical parts), and may have branching (forking or joining) configurations. Arrows do not represent flow or sequence as in the traditional process flow model. Arrows convey data or objects related to functions to be performed. The functions receiving data or objects are constrained by the data or objects made available.

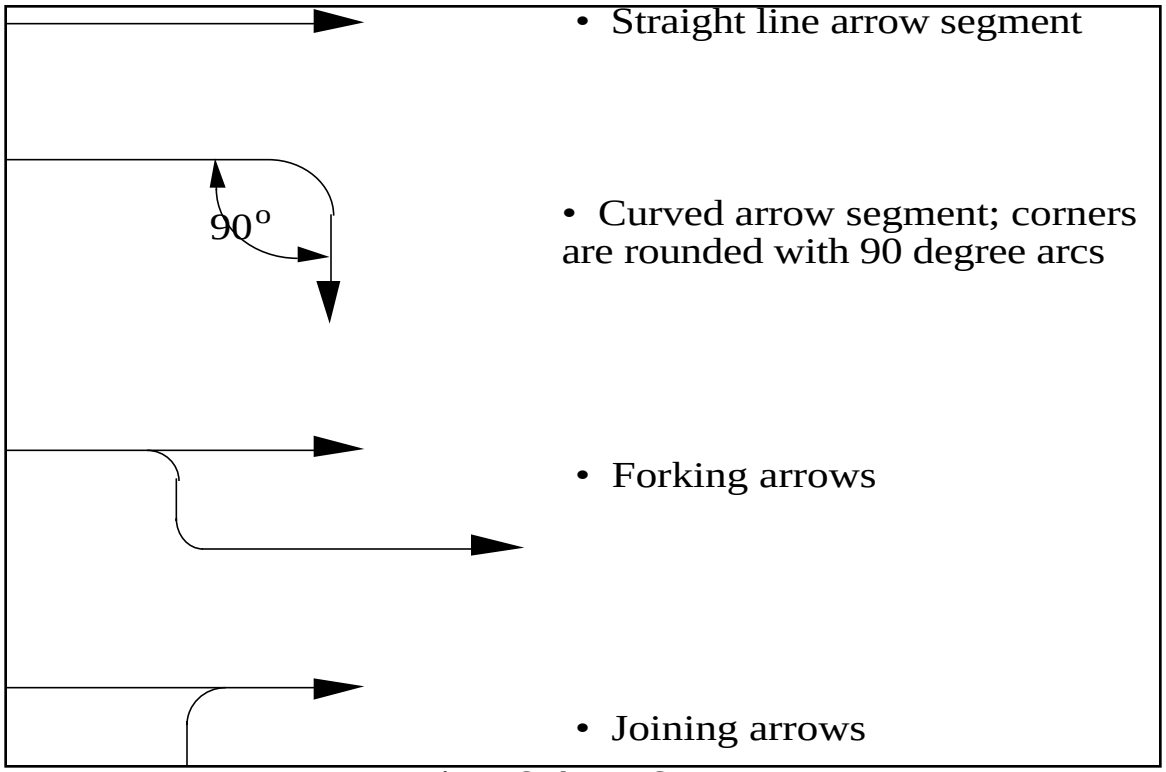


Figure 2. Arrow Syntax

Boxes

1. Boxes shall be sufficient in size to insert box name.
2. Boxes shall be rectangular in shape, with square corners.
3. Boxes shall be drawn with solid lines.

Arrows

1. Arrows that bend shall be curved using only 90 degree arcs.
2. Arrows shall be drawn in solid line segments.
3 Arrows shall be drawn vertically or horizontally, not diagonally.
4. Arrow ends shall

touch the outer perimeter of the function box and shall not cross into the box.

5. Arrows shall attach at box sides, not at corners.

Semantics refers to the meaning of syntactic components of a language and aids correctness of interpretation. Interpretation addresses items such as box and arrow notation and functional relationship interfaces.

Since IDEF0 supports function modeling, the box name shall be a verb or verb phrase, such as “Perform Inspection”, that is descriptive of the function that the box represents. The example "Perform Inspection" function transforms uninspected parts into inspected parts. The definitive step beyond the phrase-naming of the box is the incorporation of arrows (matching the orientation of the box sides) that complement and complete the expressive power (as distinguished from the representational aspect) of the IDEF0 box.

Standard terminology shall be used to ensure precise communication. Box meanings are named—descriptively—with verbs or verb phrases and are split and clustered in decomposition diagramming. Arrow meanings are bundled and unbundled in diagramming and the arrow segments are labeled with nouns or noun phrases to express meanings. Arrow-segment labels are prescriptive, constraining the meaning of their segment to apply exclusively to the particular data or objects that the arrow segment graphically represents. Arrow meanings are further expressed through fork and join syntax.

Each side of the function box has a standard meaning in terms of box/arrow relationships. The side of the box with which an arrow interfaces reflects the arrow's role. Arrows entering the left side of the box are inputs. Inputs are transformed or consumed by the function to produce outputs. Arrows entering the box on the top are controls. Controls specify the conditions required for the function to produce correct outputs. Arrows leaving a box on the right side are outputs. Outputs are the data or objects produced by the function.

Arrows connected to the bottom side of the box represent mechanisms. Upward pointing arrows identify some of the means that support the execution of the function. Other means may be inherited from the parent box. Mechanism arrows that point downward are call arrows. Call arrows enable the sharing of detail between models (linking them together) or between portions of the same model. The called box provides detail for the caller box (see Section 3.3.2.10).

Standard arrow positions are shown in Figure 3.

Supporting information concerning the function and its purpose shall be addressed in the text associated with the diagram. A diagram may or may not have associated text. When acronyms, abbreviations, key words, or phrases are used, the fully defined term(s) shall be

provided in the glossary.

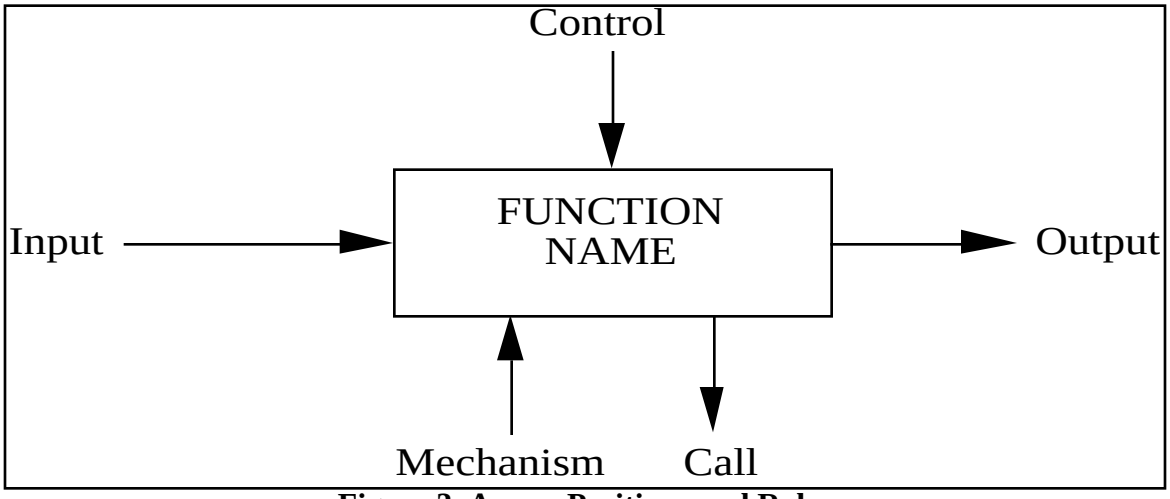


Figure 3. Arrow Positions and Roles

Boxes represent functions that show what must be accomplished. A function name shall be an active verb or verb phrase, such as:

process parts	plan resources	conduct review
monitor performance	design system	provide maintenance
develop detail design	fabricate component	inspect part

The arrows identify data or objects needed or produced by the function. Each arrow shall be labeled with a noun or noun phrase, such as:

specifications	test report	budget
design requirements	detail design	directive
design engineer	board assembly	requirements

An example depicting the placement of arrow labels and box names is shown in Figure 4.

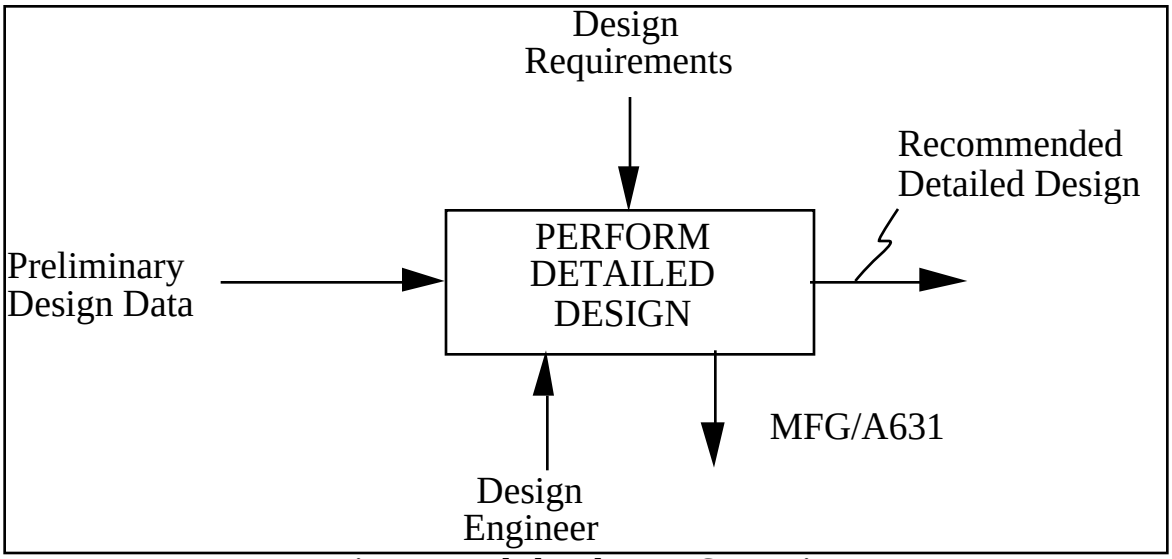


Figure 4. Label and Name Semantics

1. A box shall be named with an active verb or verb phrase.

2. Each side of a function box shall have a standard box/arrow relationship:
   a) Input arrows shall interface with the left side of a box.
   b) Control arrows shall interface with the top side of a box.

c) Output arrows shall interface with the right side of the box.

d) Mechanism arrows (except call arrows) shall point upward and shall connect to the bottom side of the box.

e) Mechanism call arrows shall point downward, shall connect to the bottom side of the box, and shall be labeled with the reference expression for the box which details the subject box.

3. Arrow segments, except for call arrows, shall be labeled with a noun or noun phrase unless a single arrow label clearly applies to the arrow as a whole.

4. A "squiggle" () shall be used to link an arrow with its associated label, unless the arrow/label relationship is obvious.

5. Arrow labels shall not consist solely of any of the following terms: function, input, control, output, mechanism, or call.

IDEF0 models are composed of three types of information: graphic diagrams, text, and glossary. These diagram types are cross-referenced to each other. The graphic diagram is the major component of an IDEF0 model, containing boxes, arrows, box/arrow interconnections and associated relationships. Boxes represent each major function of a subject. These functions are broken down or decomposed into more detailed diagrams, until the subject is described at a level necessary to support the goals of a particular project. The top-level diagram in the model provides the most general or abstract description of the subject represented by the model. This diagram is followed by a series of child diagrams providing more detail about the subject.

Each model shall have a top-level context diagram, on which the subject of the model is represented by a single box with its bounding arrows. This is called the A-0 diagram (pronounced A minus zero). The arrows on this diagram interface with functions outside the subject area to establish model focus. Since a single box represents the whole subject, the descriptive name written in the box is general. The same is true of the interface arrows since they also represent the complete set of external interfaces to the subject. The A-0 diagram also sets the model scope or boundary and orientation. An example A-0 diagram is shown in Figure 5.

The A-0 context diagram also shall present brief statements specifying the model's viewpoint and purpose, which help to guide and constrain the creation of the model. The viewpoint determines what can be "seen" within the model context, and from what perspective or "slant". Depending on the audience, different statements of viewpoint may be adopted that emphasize different aspects of the subject. Things that are important in one viewpoint may not even appear in a model presented from another viewpoint of the same subject.

The statement of purpose expresses the reason why the model is created and actually determines the structure of the model. The most important features come first in the hierarchy, as the whole top-level function is decomposed into sub-function parts that compose it, and those parts, in turn, are further decomposed until all of the relevant detail of the whole viewpoint is adequately exposed. Each sub-function is modeled individually by a box, with parent boxes detailed by child diagrams at the next lower level. All child diagrams must be within the scope of the top-level context diagram.

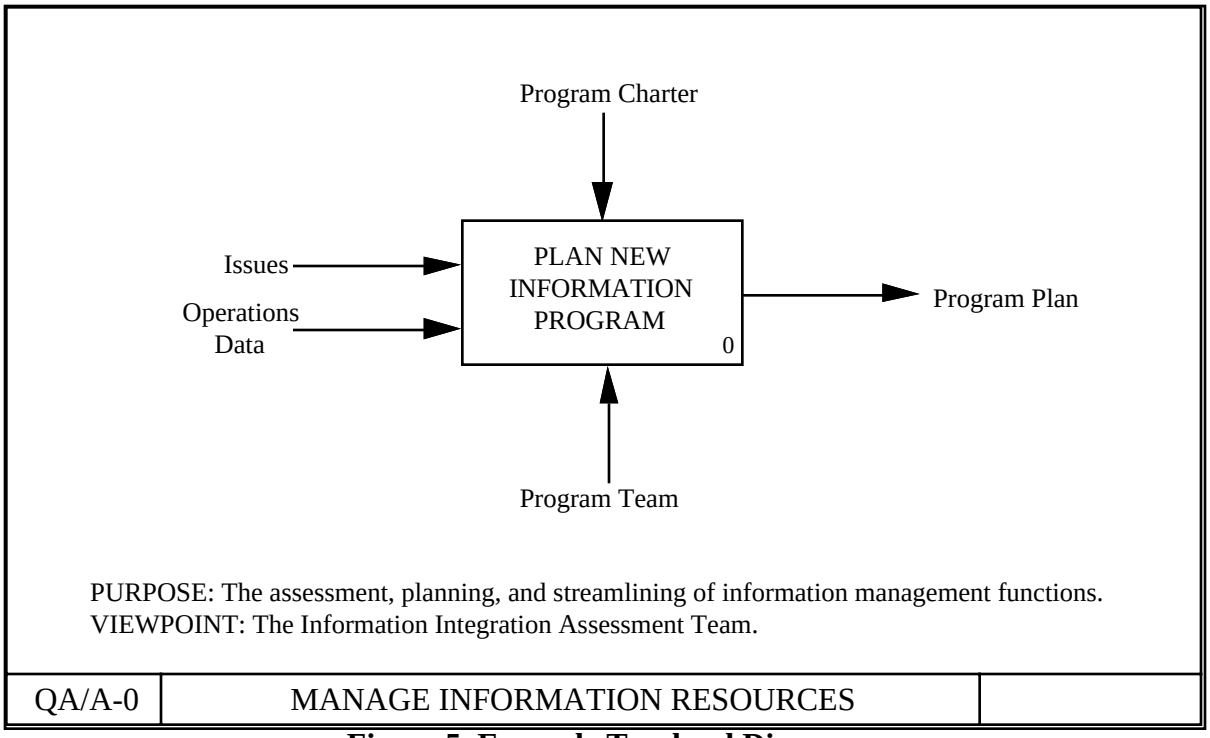


Figure 5. Example Top-level Diagram

The single function represented on the top-level context diagram may be decomposed into its major sub-functions by creating its child diagram. In turn, each of these sub-functions may be decomposed, each creating another, lower-level child diagram. On a given diagram, some of the functions, none of the functions or all of the functions may be decomposed. Each child diagram contains the child boxes and arrows that provide additional detail about the parent box.

The child diagram that results from the decomposition of a function covers the same scope

as the parent box it details. Thus, a child diagram may be thought of as being the "inside" of its parent box. This structure is illustrated in Figure 6.

A parent diagram is one that contains one or more parent boxes. Every ordinary (non-context) diagram is also a child diagram, since by definition it details a parent box. Thus a diagram may be both a parent diagram (containing parent boxes) and a child diagram (detailing its own parent box). Likewise, a box may be both a parent box (detailed by a child diagram) and a child box (appearing on a child diagram). The primary hierarchical relationship is between a parent box and the child diagram that details it.

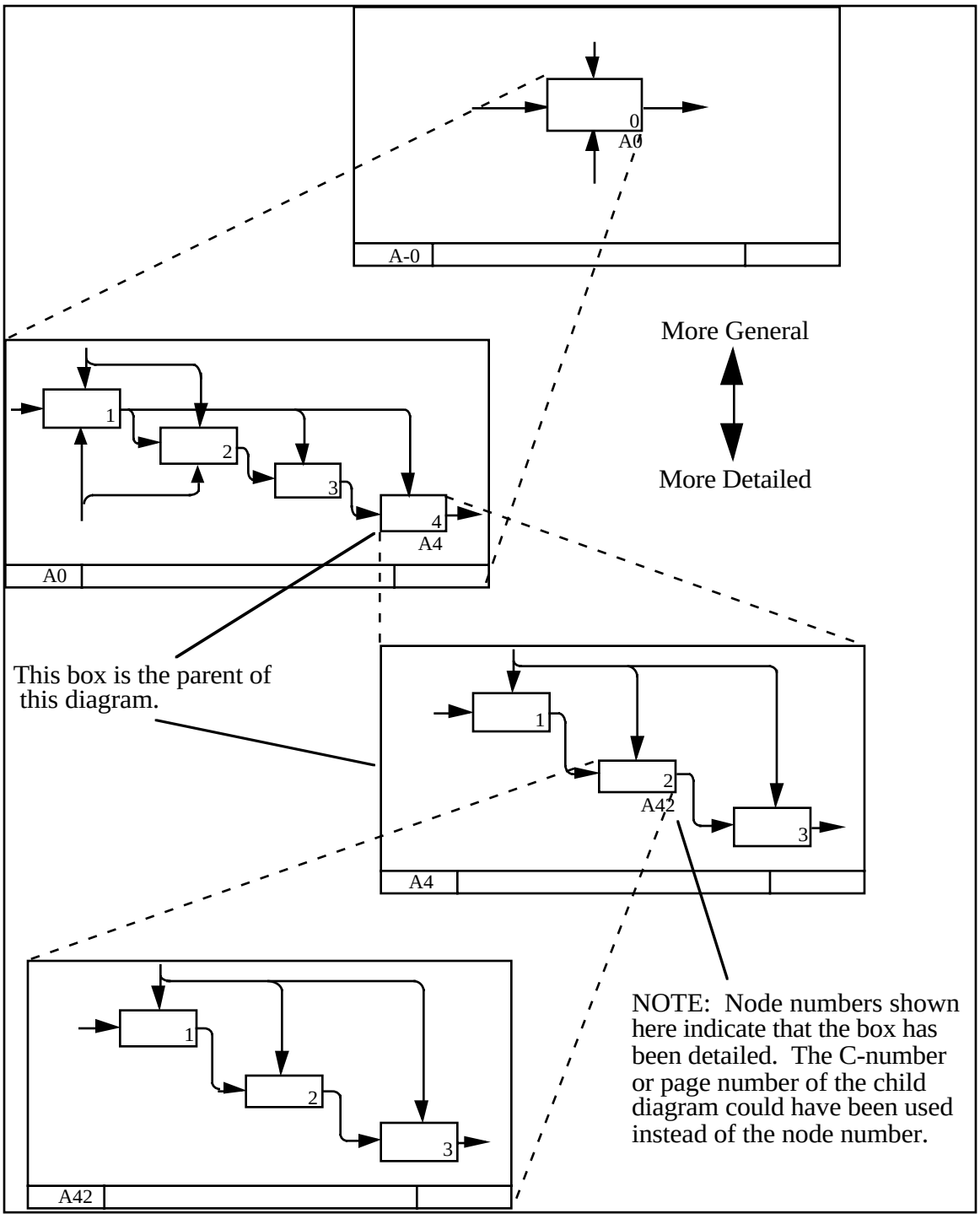


Figure 6. Decomposition Structure

The fact that a child box is detailed, and is therefore also a parent box, is indicated by the presence of a Detail Reference Expression (DRE). The DRE is a short code written below the lower right corner of the detailed (parent) box that points to its child diagram.

The DRE shall take one of the following forms:

1. A chronological creation number called a "C-number" that shall uniquely identify a particular version of a child diagram.

2. A page number of the child diagram in the published document in which the model appears.

3. The node number referencing the child diagram. If there are multiple versions of a child diagram, a particular version cannot be specified.

4. A model note number whose text specifies the conditions for selection of a particular child version.

Figure 7 illustrates the use of node numbers as DREs. In the figure, the presence of DREs under boxes 1, 2 and 3 indicates that they have been detailed on the specified child diagrams.

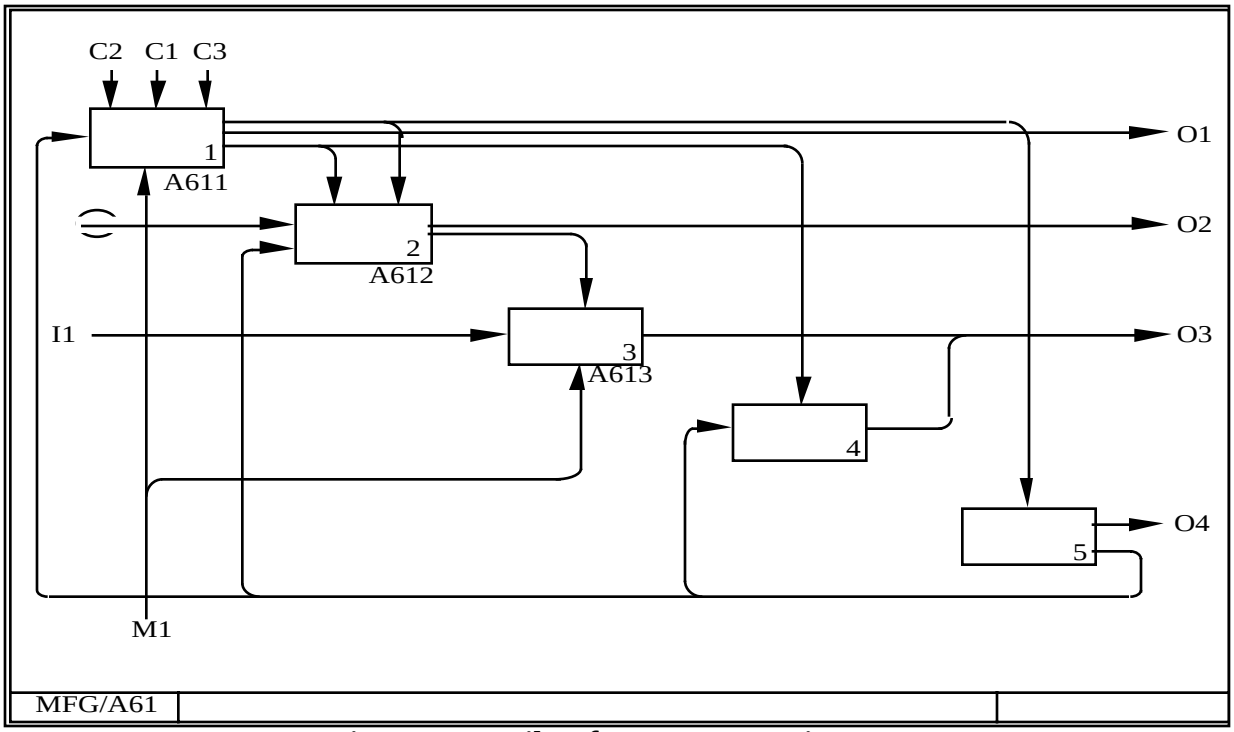


Figure 7. Detail Reference Expression Use

A diagram may have associated structured text, which is used to provide a concise overview of the diagram. Text shall be used to highlight features, flows, and inter-box connections to clarify the intent of items and patterns considered to be of significance. Text shall not be used merely to describe, redundantly, the meaning of boxes and arrows.

The glossary shall be used to define acronyms and key words and phrases that have been used in conjunction with diagram graphics. The glossary defines words in the model that must convey a common understanding in order to correctly interpret the model content.

For Exposition Only (FEO, pronounced fee-oh) diagrams shall be used where an additional level of supplementary knowledge is required to adequately understand specific areas of a model. Supplementary detailing should be limited to what is needed to achieve the stated purpose for a knowledgeable audience. A FEO diagram need not comply with IDEF0 syntax rules.

Arrows on an IDEF0 diagram represent data or objects as constraints. Only at low levels of detail can arrows represent flow or sequence, when the modeled subject is sufficiently detailed to treat specific changes made to specific data items or objects. Connecting the output of one box to the input, control, or mechanism of another box shows that the function modeled by the latter box requires, and thus is constrained by, the presence of the corresponding output of the former box. This type of constraint is illustrated in Figure 8. The arrows connecting to a box represent all the data and objects that are needed for the function to be completely performed.

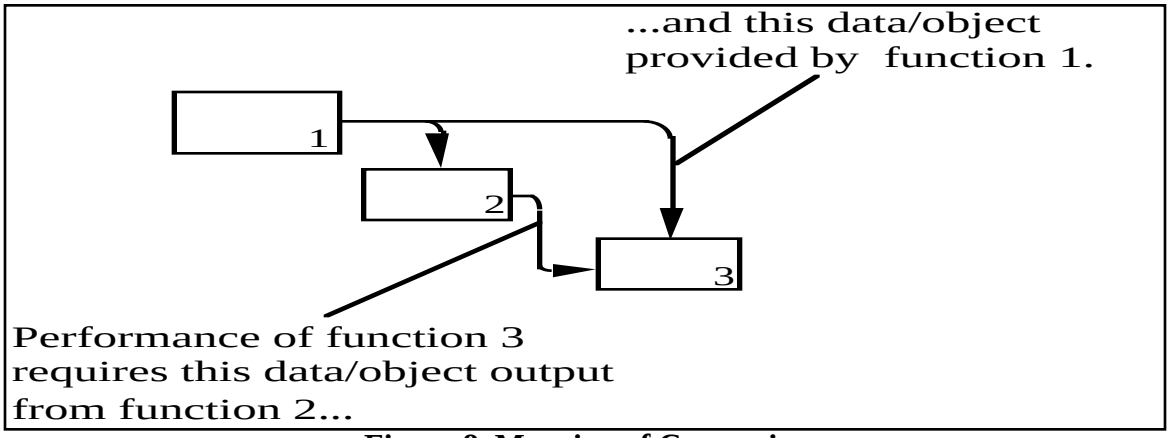


Figure 8. Meaning of Constraint

A box may perform various parts of its function under different circumstances, using dif-

ferent combinations of its input and controls and producing different outputs. These different performances are called the different activations of the box.

Several functions in a model may be performed concurrently, if the needed constraints have been satisfied. As illustrated in Figure 9, an output of one box may provide some or all of the data and objects needed for activations of one or more other boxes.

When an output of one box provides some or all of the inputs, controls or mechanisms needed by another box, a given activation of the latter box may depend on sequential performance. However, different activations of the same box(es), with possibly different requirements, may operate concurrently.

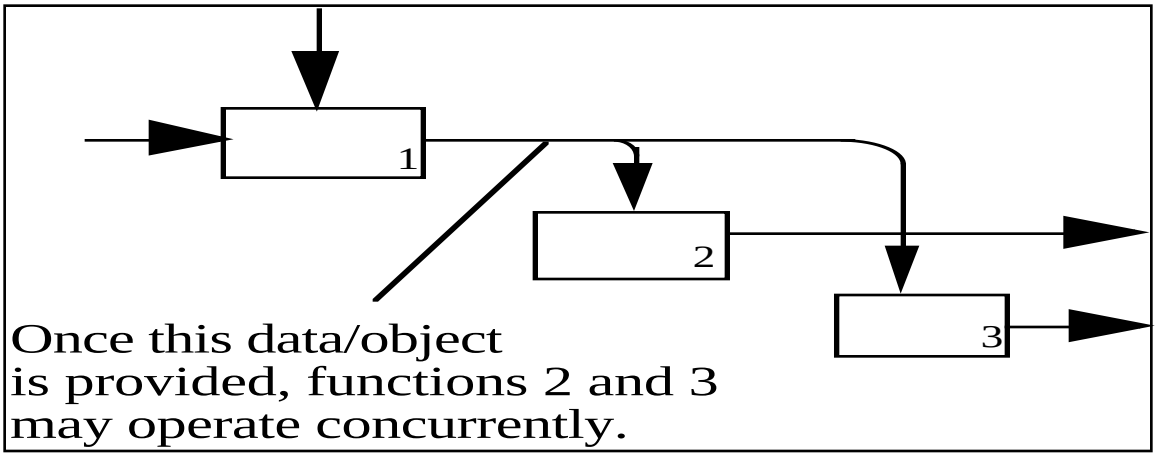


Figure 9. Concurrent Operation

It is useful to think of high-level arrows as pipelines or conduits. High-level arrows have general labels, while arrows on lower-level diagrams have more specific labels. If an arrow splits, forming two or more new arrow segments, each arrow segment may have a more specific label, as shown in Figure 10.

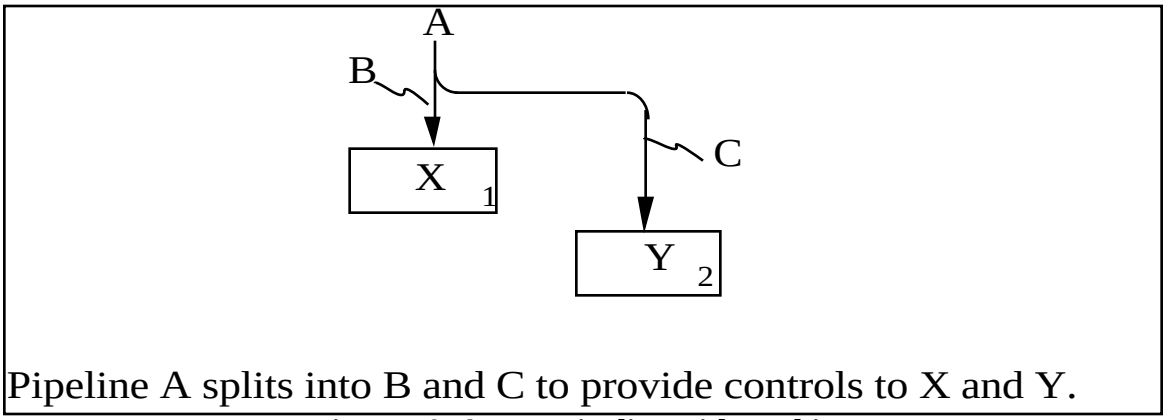


Figure 10. Arrow Pipeline with Forking

An arrow may branch (fork or join), indicating that the same kind of data or object may be needed or produced by more than one function. The branches may represent either the same thing or portions of the same thing. Since labels specify what arrow segments represent, labels on branching arrow segments provide a detailing of the arrow content just as lower level diagrams provide detailing of parent boxes.

All or part of the contents of an arrow may follow a branch. A forking arrow may denote the "unbundling" of meanings that had been combined under a more general label. The joining of two arrow segments may denote "bundling", i.e., the combining of separate meanings into a more general category. All contents are provided through all branches unless otherwise indicated by a special label on each arrow segment. These conventions are illustrated in Figure 11.

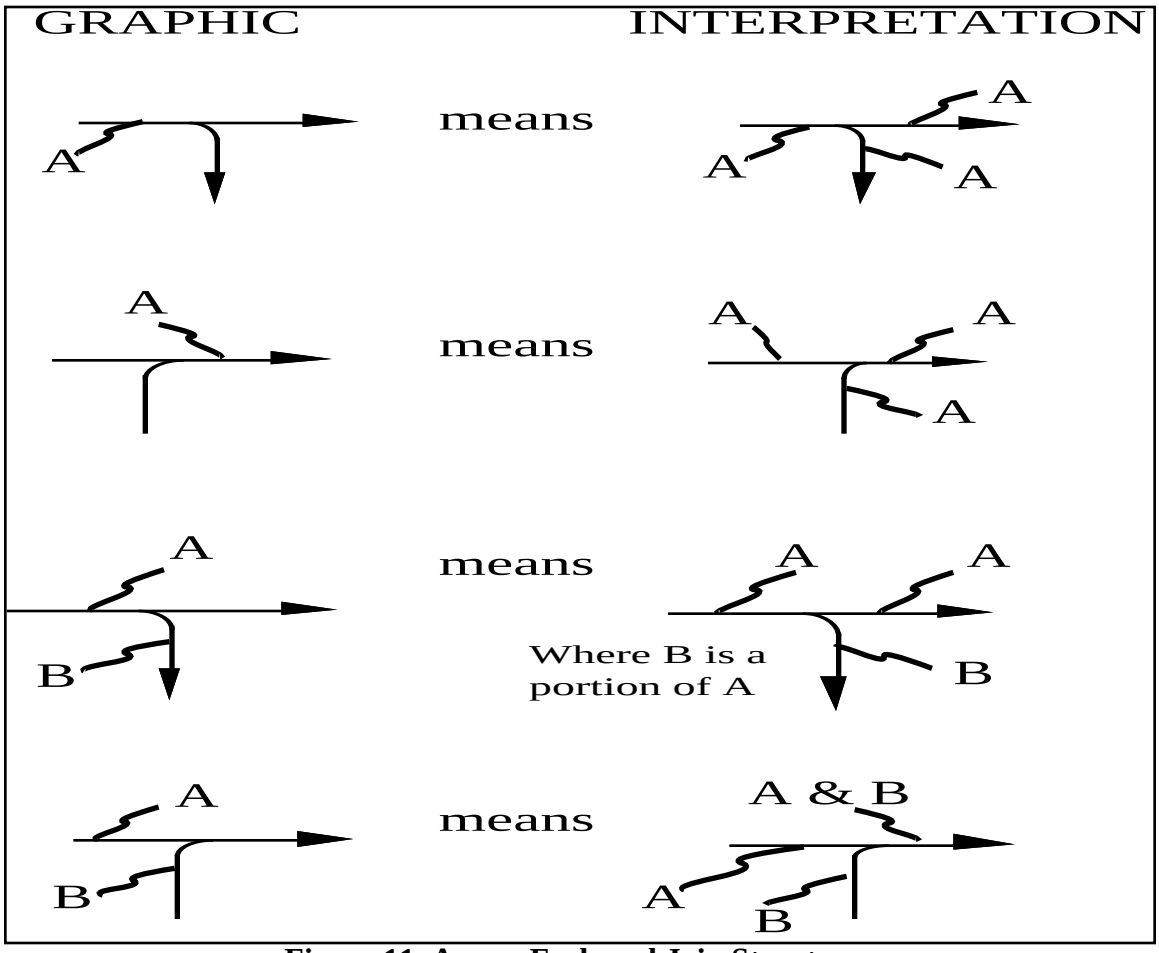


Figure 11. Arrow Fork and Join Structures

Except for the single-box A-0 context diagram, a graphic diagram contains a minimum of three and a maximum of six boxes. Boxes are normally organized diagonally from the upper left corner to the lower right, i.e., in a "staircase" configuration.

Any output arrow may provide some or all of the input, control, or mechanism data or objects to any other box. An output arrow may provide data or objects to several boxes via the forking mechanism, as shown in Figure 12.

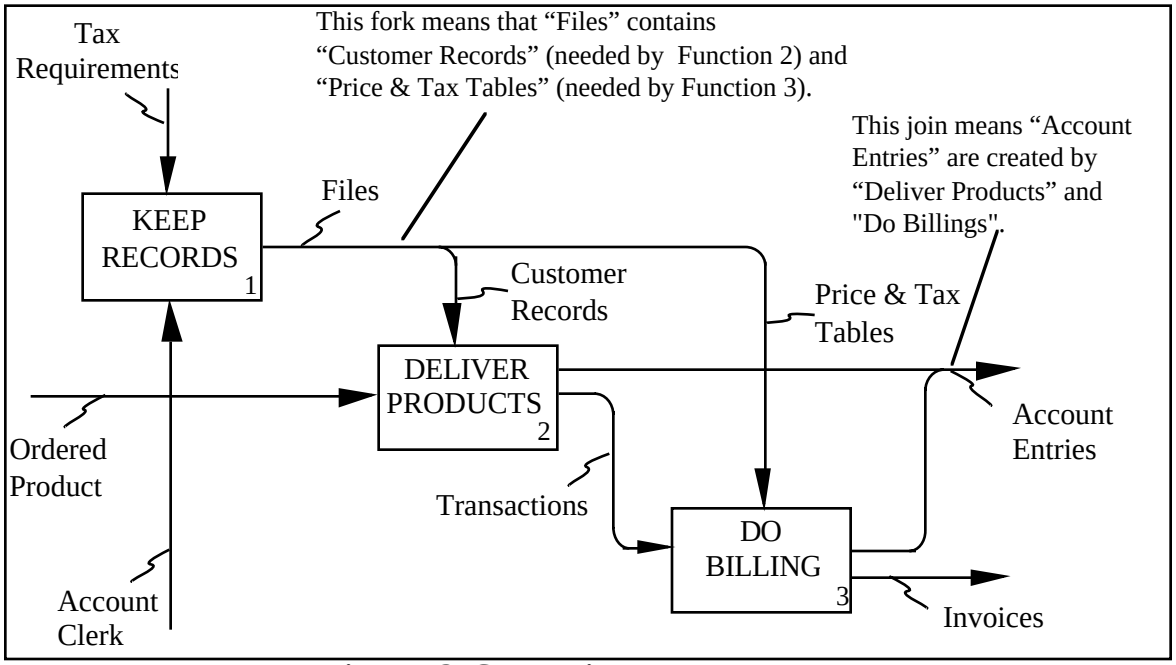


Figure 12. Connections Between Boxes

If a box on a diagram is detailed by a child diagram, each arrow connected to the parent box shall appear on the child diagram, unless the arrow is tunneled next to its parent box (see Section 3.3.2.9).

On a diagram, data or objects may be represented by an internal arrow, with both ends (source and use) connected to boxes, or by a boundary arrow, with only one end (source or use) connected. Internal arrows and boundary arrows are shown in Figure 13. Boundary arrows are discussed in detail in Sections 3.3.2.7 and 3.3.2.8.

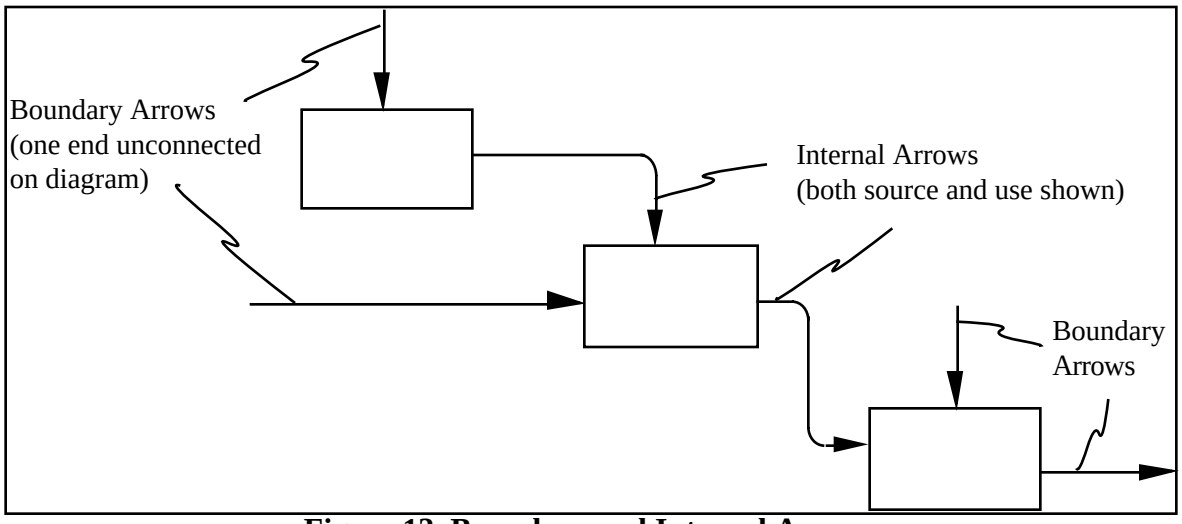


Figure 13. Boundary and Internal Arrows

Boundary arrows on an ordinary (non-context) graphic diagram represent the inputs, controls, outputs, or mechanisms of the diagram's parent box. The source or use of these boundary arrows can be found only by examining the parent diagram. All boundary arrows on a child diagram (except for tunneled arrows, Section 3.3.2.9) shall correspond to the arrows that connect to its parent box, as shown in Figure 14.

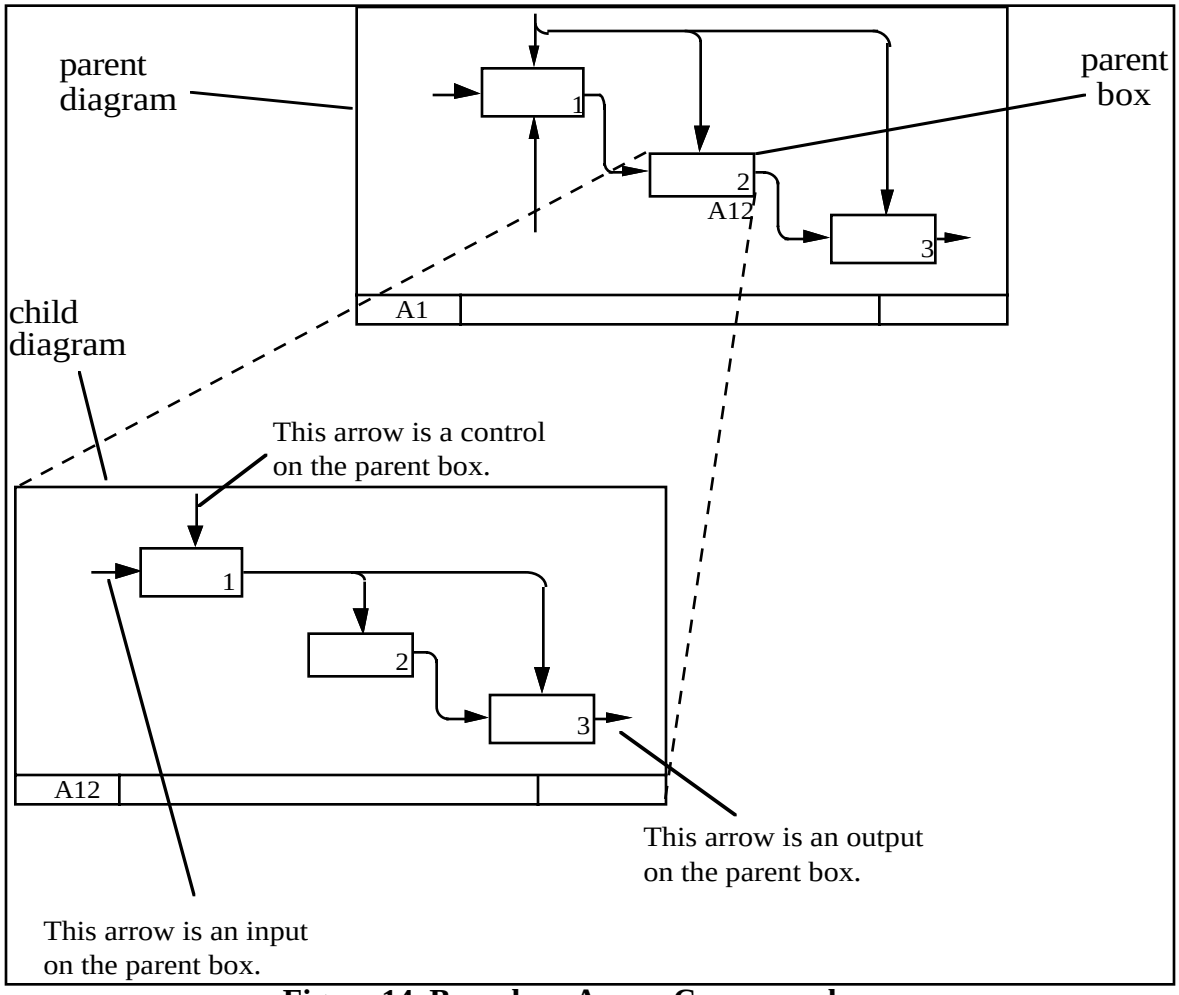
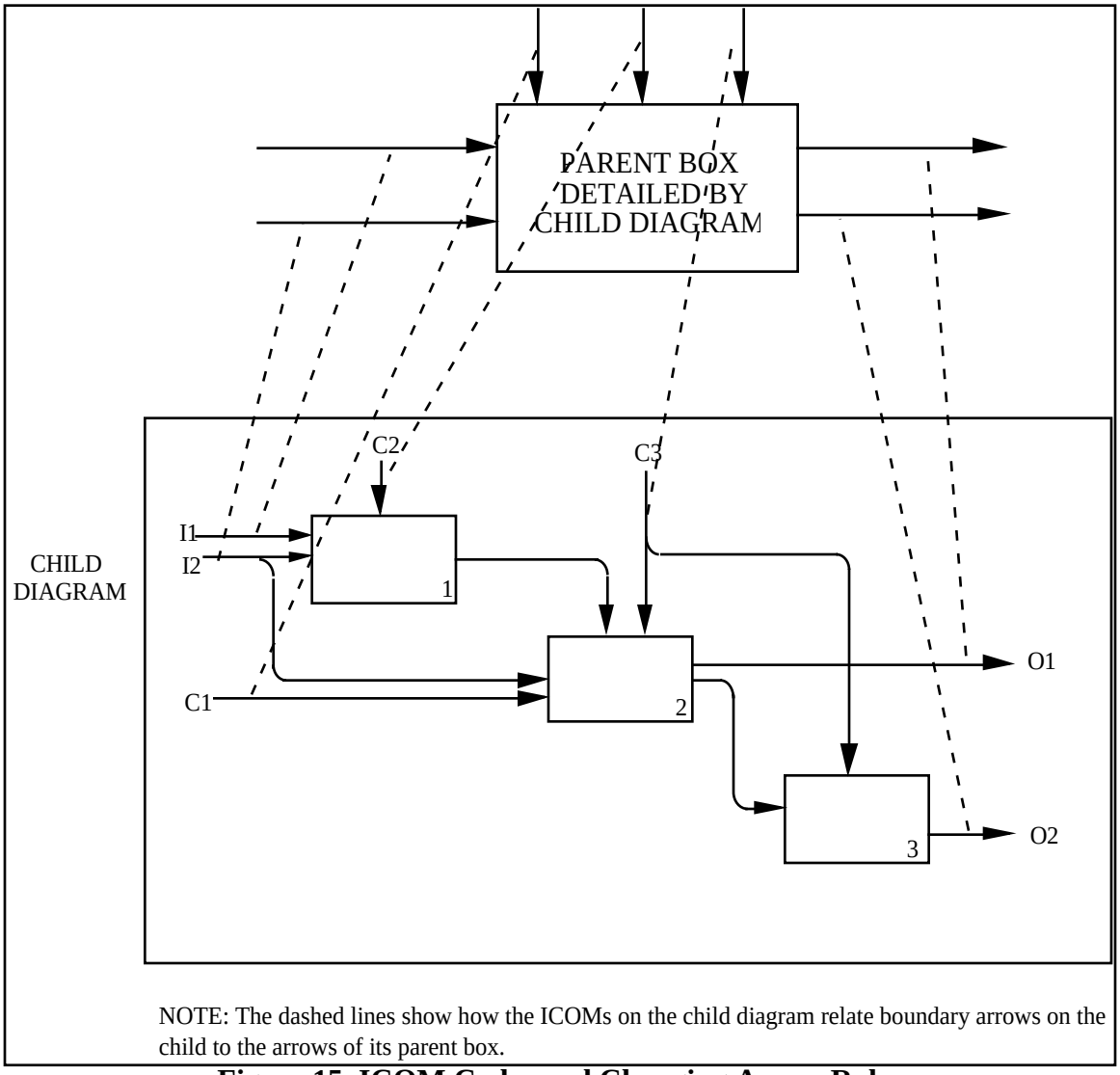


Figure 14. Boundary Arrow Correspondence

ICOM codes relate boundary arrows on a child diagram to arrows connected to its parent box. A specific notation, called ICOM codes, specifies the matching connections. The letter I, C, O or M is written near the unconnected end of each boundary arrow on the child diagram. This coding identifies the arrow as an Input, Control, Output or Mechanism on the parent box. This letter is followed by a number giving the relative position at which the arrow is shown connecting to the parent box, numbering from left to right or top to bottom. For example, “C3” written on a boundary arrow on a child diagram indicates that this arrow corresponds to the third control arrow (from the left) entering its parent box.

This coding relates each child diagram to its own immediate parent box. If boxes on a child diagram are detailed on subsequent child diagrams, new ICOM codes are assigned on each new child diagram, relating that diagram's boundary arrows to arrows on its own immediate parent box.

Using the letter-numbering matching scheme of ICOM coding, arrow roles (input, control, mechanism) may differ between parent and child diagrams. Figure 14 shows the common case where the roles do match, e.g., the input to the parent box matches the input on the child diagram. As an example of changing roles, a control arrow on a parent box may be either an input or a control arrow for boxes on its child diagram. Likewise, a control for a parent box may be an input for one or more of its child boxes. Figure 15 shows examples of changing arrow roles.



NOTE: The dashed lines show how the ICOMs on the child diagram relate boundary arrows on the child to the arrows of its parent box.

Figure 15. ICOM Codes and Changing Arrow Roles

A tunneled arrow is used to provide information at a specific level of decomposition that is not required for understanding at some other levels. An arrow can be tunneled at any chosen level.

Using the parentheses notation illustrated in Figure 16, tunneling an arrow where it connects to a box side means that the data or objects expressed by that arrow are not necessary for understanding subsequent level(s) of decomposition, and thus shall not be shown on its child diagram. However, because this arrow does correspond to one on its parent diagram, it is given an ICOM code. This code may be used elsewhere in the model, e.g., in a reference expression on a diagram where the arrow reappears, in order to identify the location of the original tunneling. An arrow tunneled at its connected end may be omitted from one or more levels of decomposition and then reappear on another level, in one or more places, tunneled at the unconnected end.

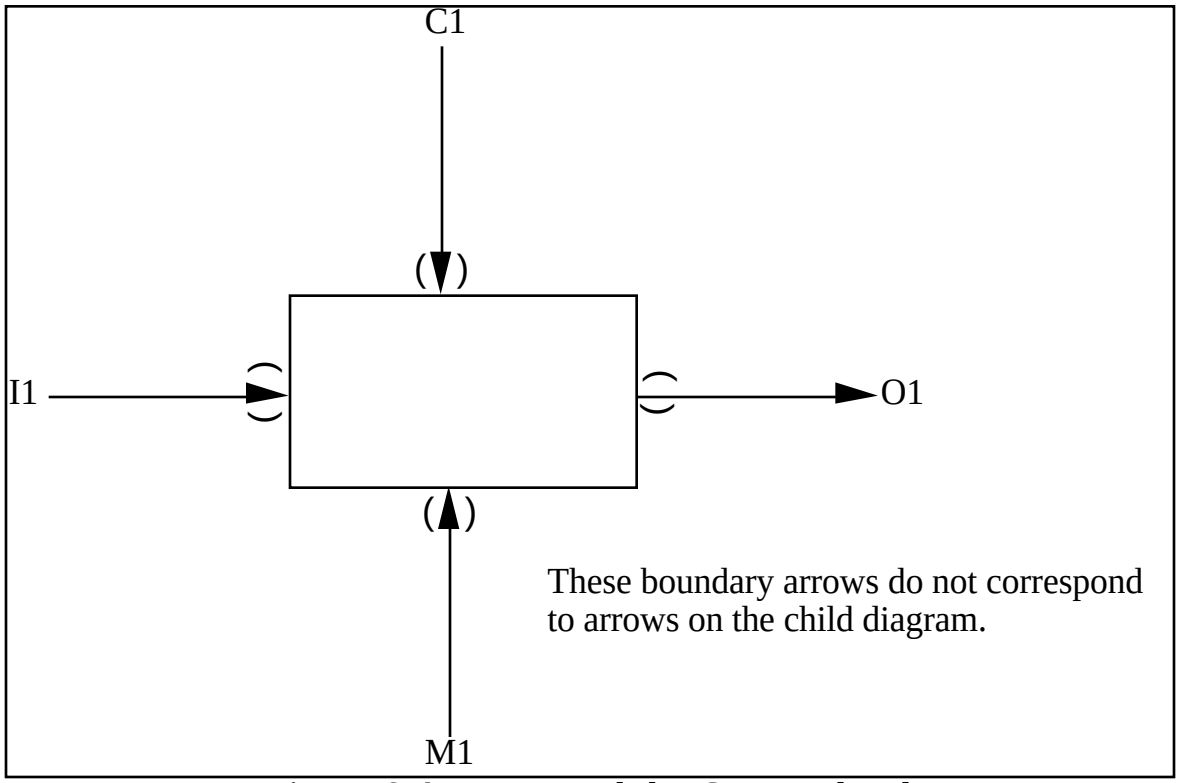


Figure 16. Arrows Tunneled at Connected End

Tunneling an arrow at the unconnected end means that the data or objects are not necessary at the next higher (parent) level and hence shall not be shown connecting to the parent box. This is shown in Figure 17. Because this arrow does not correspond to one on its parent diagram, it does not have an ICOM code. The arrow may have an attached model note containing the node reference and ICOM code that locates the "other end" of the tunnel. ICOM coding for the arrow resumes for any subsequent child diagrams.

Figure 18 provides an example of tunneled arrows on parent and child diagrams.

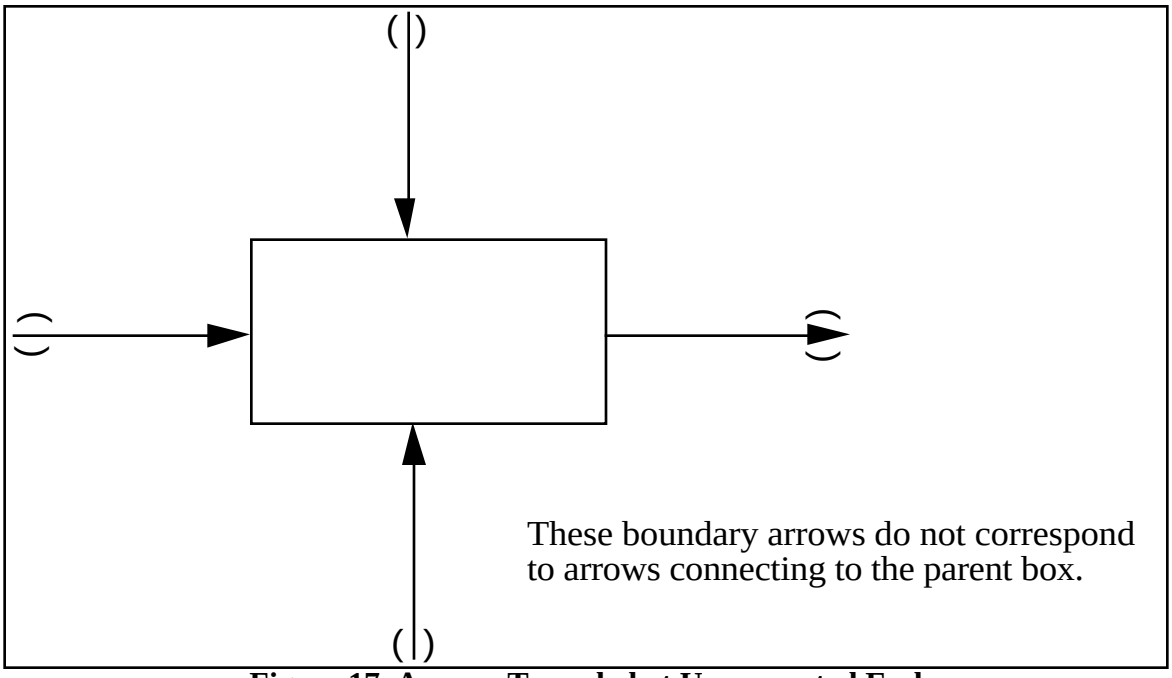


Figure 17. Arrows Tunneled at Unconnected End

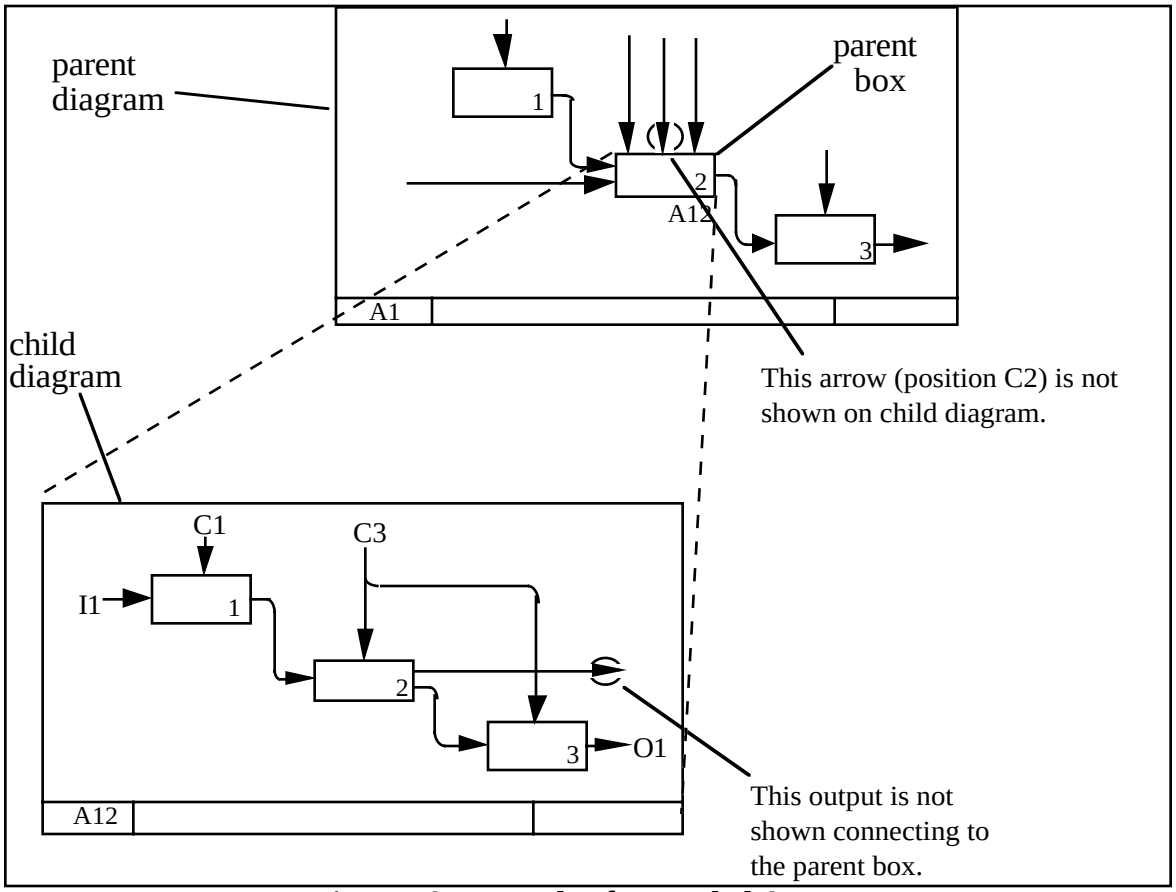


Figure 18. Example of Tunneled Arrows

A call arrow is a special case of mechanism arrow. It signifies that the caller box does not have its own child diagram to detail it, but rather is detailed entirely by another box (and its descendants) in the same or another model. Multiple caller boxes may call the same box.

The call arrow is labeled with the node reference of the diagram containing the called box, along with the called-box number. A caller box may call only one box in a given activation. However, depending on conditions specified in a model note attached to a call arrow, the caller box may select one of several possible called boxes. In this case, the call arrow label shall include a list of the node references of all the possible called boxes.

The arrows of the called box may not correspond exactly with those of the caller box, either in number or in meaning. In these cases, model notes attached to the call arrows shall specify the relationships so that the correct interpretation may be given to the shared data and objects.

1. Context diagrams shall have node numbers A-n, where n is greater than or equal to zero.

2. The model shall contain a A-0 context diagram, which contains only one box.

3. The box number of the single box on the A-0 context diagram shall be 0.

4. A non-context diagram shall have at least three boxes and no more than six boxes.

5. Each box on a non-context diagram shall be numbered in its lower right inside corner, in order (from upper left to lower right on the diagram) from 1 to at most 6.

6. Each box that has been detailed shall have the detail reference expression (DRE, e.g., node number, C-number, or page number) of its child diagram written beneath the lower right corner of the box.

7. Arrows shall be drawn as horizontal and vertical straight line segments. Diagonal line segments shall not be used.

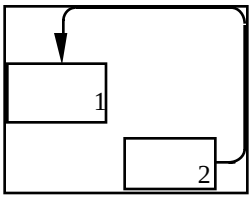
8. Each box shall have a minimum of one control arrow and one output arrow.

9. A box shall have zero or more input arrows.

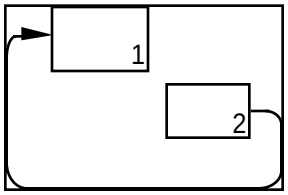
10. A box shall have zero or more non-call mechanism arrows.

11. A box shall have 0 or 1 call arrows.

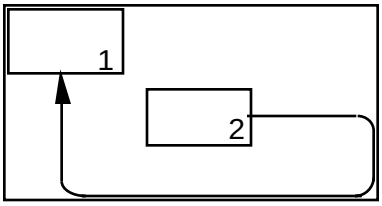
12. Control feedbacks shall be shown as “up and over”.



Input feedbacks shall be shown as “down and under”.



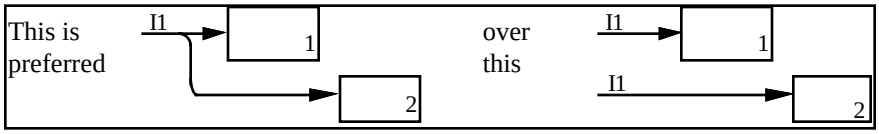
Mechanism feedbacks shall be shown as “down and under”.



13. The unconnected end of a boundary arrow shall have the proper ICOM code specifying its connection to the parent box, or shall be tunneled.

14. Open-ended boundary arrows that represent the same data or objects shall be connected through a fork to show all the places affected, unless this results in an unreadable diagram. Multiple sources that represent the same data or objects

shall join to form a single output boundary arrow.



15. Box names and arrow labels shall not consist solely of the following words: function, activity, process, input, output, control or mechanism.

Reference expressions use codes that are assigned to model features such as diagrams, boxes, arrows and notes. Reference expressions then can be used in various contexts to refer precisely to any aspect of the model.

The basic unit of reference is the node number, which applies to the place where functional decomposition is modeled by the detailing of a parent box on a child diagram. All other reference codes are based on node numbers.

Each box on a diagram shall be numbered in the lower inside right corner of the box. This numbering system is required to uniquely identify the boxes within a diagram, and to generate node numbers. It is also used to cross-reference descriptive entries in the text and glossary to the boxes on a diagram.

The box number for the single box on the A-0 context diagram shall be 0 (zero). The box numbers for the boxes on all other graphic diagrams shall be 1, 2, 3, to at most 6, to uniquely identify the three to six boxes on each such diagram. For boxes arranged diagonally on the diagram from the top left corner to the bottom right corner, box numbers are assigned in order, starting at the upper left. If off-diagonal boxes are also used, the numbering sequence starts with the on-diagonal boxes and then continues, from the lower right, in counter-clockwise order.

A node number is based on the position of a box in the model hierarchy. Normally, a node number is formed by appending a box number to the node number of the diagram on which it appears. For example, the node number of box 2 on diagram A25 is A252. (All IDEF0 node numbers begin with a capital letter, such as "A".) When a box is detailed by a child diagram, the node number of the parent box is assigned as the diagram node number; thus, the parent box and its child diagram have the same node number.

Context diagrams and the top-level child diagram are exceptions to the above node-numbering scheme. Every IDEF0 model has a top-level context diagram, the A-0 diagram. This diagram contains a single "top box" which is the unique parent of the entire modeled

subject and bears the unique box number 0 (zero) and node number A0. Every IDEF0 model shall also have at least three, but no more than six, child boxes on the A0 child diagram that details the A0 parent top box, those boxes bearing the unique node numbers A1, A2, A3, to at most A6. Thus, the sequence [A0, A1, ..., A2, ..., A3, ...] starts the node numbering for each model.

For example, a model might have the following node numbers:

... Optional higher-level context diagrams
A-1Optional context diagram
A-0Required top-level context diagram (contains A0 top box)
A0 Top level child diagram
A1, A2, ..., A6 Child diagrams
A11, A12, ..., A16, ..., A61, ... , A66 Child diagrams
A111, A112, ..., A161, ..., A611, ..., A666Child diagrams
... Lower-level child diagrams

Node numbers may also be used as detail reference expressions to indicate the detailing of a parent box by a child diagram. If a function has been decomposed, the node number of the child diagram which detailed it may be written beneath the lower right corner of the parent box. In Figure 7, the DREs (in this case, node numbers) for boxes 1, 2 and 3 indicate that they have been detailed and identify the child diagrams.

The node index is a presentation of node information in an "outline" format. All node numbers, along with either diagram titles or box names, shall be presented in an indented form that exhibits the nested hierarchic structure of the model. This places related diagrams together in the order used in an ordinary Table of Contents, as is illustrated in Figure 19.

A0 Manufacture Product

A1 Plan For Manufacture

A11 Identify Manufacturing Methods
A12 Estimate Requirements, Time, Cost to Produce
A13 Develop Production Plans
A14 Develop Support Activities Plan

A2 Make and Administer Schedules and Budgets

A21 Develop Master Schedule
A22 Develop Coordinating Schedule
A23 Estimate Costs & Make Budgets
A24 Monitor Performance To Schedule & Budget

A3 Plan Production

Figure 19. Typical Node Index

The developed IDEF0 model with its structured decomposition provides the basis to sketch the full decomposition in node tree fashion on a single large diagram. The use of a node tree is optional. The content of the node tree shall be identical to that of the node index or any suitable portion of interest.

There is no standard format for the actual display of the node information, except that the hierarchy shall be shown graphically as a tree rooted at a chosen node (e.g., A0 for the whole model). Figure 20 provides an illustration.

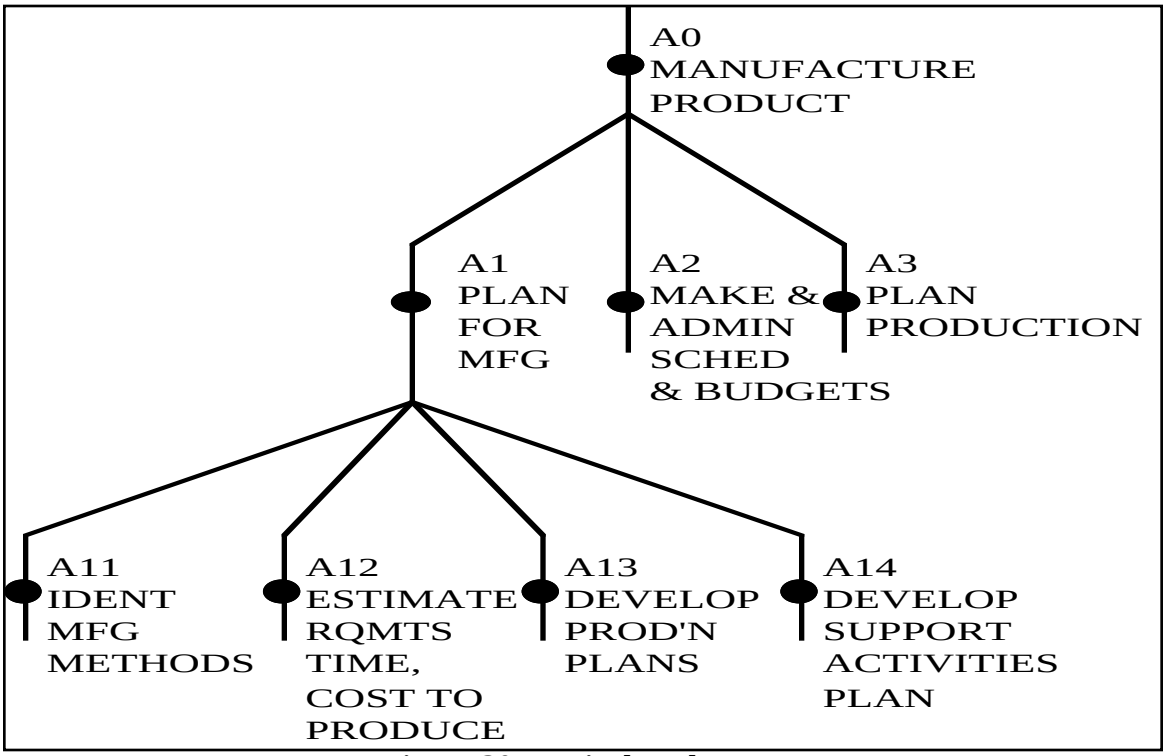


Figure 20. Typical Node Tree

Each diagram in a model has a node reference, which uniquely identifies it and its position in the model hierarchy. The node reference is composed of the abbreviated model name (see Section 3.4.5) and the diagram node number (see Section 3.3.4.2), separated by a slash (/). For example, a model named Quality Assurance Operations might be abbreviated as QA, and a node reference might then be QA/A312. References to a diagram in the same model may omit the model name abbreviation, using only the diagram node number.

A node reference may also have a suffix, e.g., F (for FEO), T (for text), or G (for glossary), and a page number. For example, a node reference for a FEO might be QA/A321F1 (see Section 3.4.4).

Model notes are optional. They are denoted by an integer "n" inside a small square box ()

For a given diagram, the note numbers shall form a consecutive sequence, starting at 1. Vertical pipes surrounding the note number (|n|), may be used as an alternative notation.

Model notes provide information that is relevant to (and an integral part of) a diagram's message, but that does not conveniently fit into the box-and-arrow syntax. If the text of the note is to apply to several places or several features of the diagram, the boxed note number (without text) may be copied and may be attached by an IDEF0 squiggle to each point of application, but only on the single diagram on which the model note's text appears.

A standard notation is used in writing text and notes to refer to diagrams and specific parts of diagrams. References are based on box numbers, node numbers, ICOM codes, and note numbers. The following table provides examples of reference notations.

REFERENCE NOTATION	MEANING
2I1	Box 2 Input 1
O2	The boundary arrow with ICOM code O2
2O2 to 3C1 or 2o2 to 3c1	The arrow from 2O2 to 3C1 (The I, O, C or M may be upper case or lower case.)
I2 to 2I3 to 2O2 to (3C1 and 4C2)	From the boundary arrow with ICOM code I2 to Box 2 Input 3, through the activation of Box 2 that yields Output 2, to the availability (via a forking branch) of that output as Control 1 on Box 3 and Control 2 on Box 4.
A21.3C2	On diagram A21 in this model, see Box 3 Control 2. An embedded period means "look specifically at".
A42. 	On diagram A42, see model note 3.
A42.|3|	Same as above, using optional notation (vertical pipes surrounding model note instead of boxed note).
A42.3	On diagram A42 in this model, see Box 3.

MFG/A42.1	On diagram A42 of the model abbreviated MFG, see Box 1.

One of the most important features of IDEF0 as a modeling concept is that it gradually introduces greater and greater levels of detail through the diagram structure comprising the model. In this way, communication is enhanced by providing the reader with a well-bounded topic with a manageable amount of detail to learn from each diagram.

An IDEF0 model starts by presenting the whole subject as a single unit – a box with external-arrow boundary conditions connecting it to functions and resources outside the subject. The single box is called the "top box" of the model. (This top box has node number A0.) Since the single top box of an IDEF0 model represents the subject as a whole, the descriptive name in the box is general. The same is true of the external arrows of the model, since they represent the complete set of external boundary conditions of the subject as a whole, including access to mechanism support that supplies additional means of performance.

The diagram in which the A0 top box appears represents the context of the model and is called a context diagram. The minimum context for a model is the special context diagram with the node number A-0. The A-0 context diagram has only the single named A0 top box, with its labeled external arrows, and also textual definitions of the Viewpoint and Purpose of the model. (The A-0 diagram has no ICOM codes or tunneling at unconnected arrow ends.)

Sometimes, however, in order to provide a more complete exposition of the environmental context of the model, an optional A-1 context diagram (having the appearance of an ordinary, non-context, diagram) is also presented. In the A-1 context diagram, the A0 box takes the place of one of the three-to-six numbered boxes (the other boxes retaining their expected box number), so the effect is to provide a complete parent diagram (with three to six boxes) for the model's top – the A1 through A6 nodes still being the first-generation children. In the case where an A-1 context diagram is used, an A-0 context diagram is still presented. This A-0 diagram still has only the single named A0 top box, with its labeled external arrows and also textual definitions of the Viewpoint and Purpose of the model.

Context diagrams are diagrams that have node numbers of the form "A-n" (with a minus sign included), where n is greater than or equal to zero. Ordinary, non-context diagrams lack the minus sign in their node numbers. The box number of the top box of the model (representing the whole of the modeled subject) always is 0. Box number 0 shall appear on the required A-0 context diagram of the model, and shall also appear on the optional A-1 context diagram (if any) where it takes the place of one of the boxes (1 to at most 6) of that A-1 (model-wide) parent diagram. Thus A0 always is the (shared) node number of the parent box and child diagram for the whole model and always is detailed by boxes with node numbers A1, A2, A3, to at most A6.

With only one box, A-0 is a proper context diagram but is not a (proper) parent diagram. Proper diagrams have three to six boxes. The parental context is that which provides or

names the context for a diagram in the place of a proper parent diagram. The parental context of the A0 diagram is the required A-0, if there is no A-1 context diagram. If there is an A-1 context diagram, A-1 is the proper parent of the A0 diagram. The parental context of the A-0 context diagram always is "TOP".

High-level context diagrams have node numbers of the form A-n, for n greater than one. Thus A-1 is a context diagram, is a proper parent (of A0), but is not high level. For a given model presentation, the highest-level context diagram (largest n) has parental context "NONE", unless the highest-level context diagram is A-0.

Each high-level context diagram, A-n, syntactically is an ordinary detail diagram except that one of its three-to-six boxes has its box number replaced by "minus sign n-1", so that, for A-1, that box is the A0 top box of the model, and the model as a whole (that A0 box itself, the parent of the children) appears to have parent A-1, grandparent A-2, etc.

By providing a more complete description of the model's environmental context (not, however, intended to be definitive in all respects, but only "typical"), context modeling (characterized by negative node numbering) provides more-constraining specifications on the boundary conditions of the A0 diagram of the model.

Context modeling proceeds just as ordinary detail modeling, the only difference being the negative numbering (and the non-definitive, but normative interpretation) that preserves A0 as the "origin" of the node-number-based coordinate system for all model references.

All the negative-node-number modeling merely provides more and more details about the sources and uses of the external boundary conditions. That detail may not precisely be matched by any particular specific environment, completely. These context diagrams describe the "typical" context.

Figure 21 provides an illustration in node-tree form to show how rich high-level context might appear.

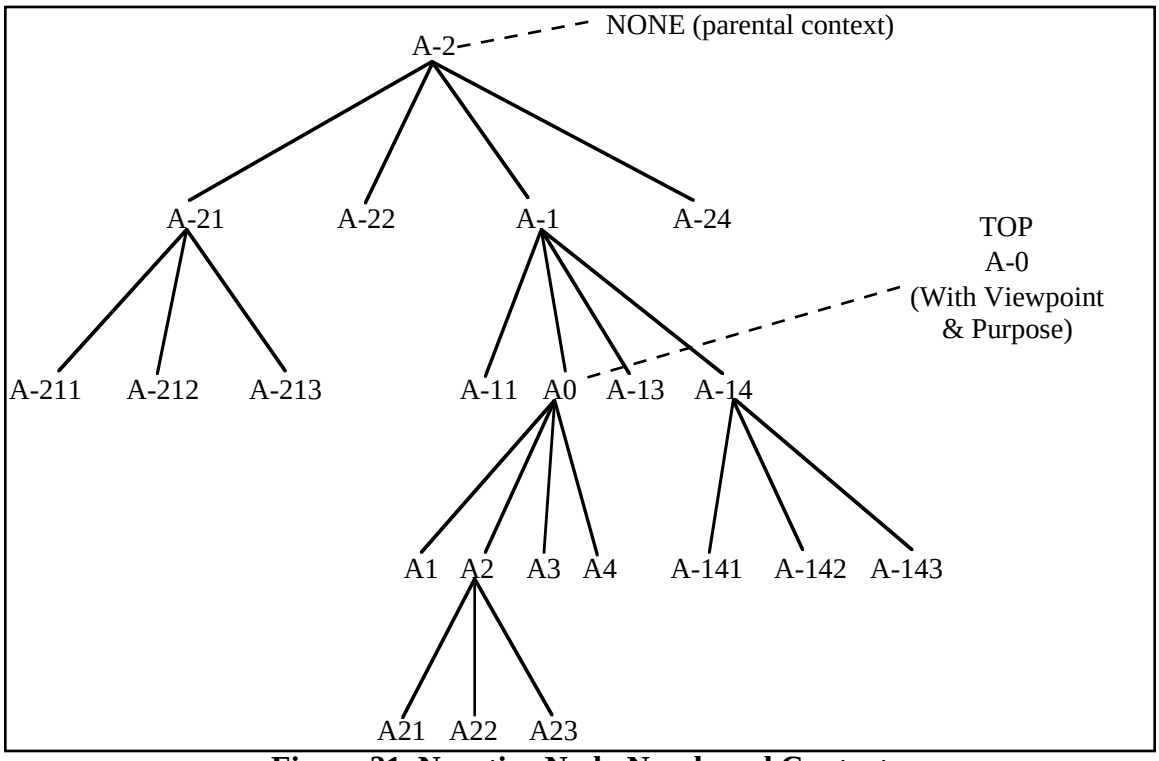


Figure 21. Negative Node-Numbered Context

The node-numbering scheme provides the basis for coordinating FEOs, text, and glossary terms. During development it is important that each new element of information be associated with the node that brought it into consideration.

For each form (FEOs, text, and glossary), the node-numbering extension notation consists of a single letter appended to the associated node number. For example, node numbers for FEOs shall contain an “F” for FEO (e.g., A312F).

Some IDEF0 users record glossary definitions on IDEF0 diagram forms, though the use of this form for glossaries is not required. In this case, a glossary page shall define the key words, phrases and acronyms used with a particular, associated IDEF0 node. The node numbers for such glossary pages shall contain a "G" for glossary (e.g., A312G).

Likewise, some IDEF0 users record their textual comments on IDEF0 diagram forms, though the use of this form for text is not required. In this case, a text page shall provide the text comments for a particular, associated IDEF0 node. The node numbers for such text pages shall contain a "T" for text (e.g., A312T).

If there is more than one FEO, glossary or text page associated with a given IDEF0 node, the pages should be designated with an additional number to uniquely identify each (e.g., A312F1, A312F2,...A312G1, A312G2,..., A312T1, A312T2, ...).

Each model has a unique, descriptive name that distinguishes it from other models with which it may be associated. This model name is normally abbreviated (uniquely) for use in node references. For example, a model named Manufacturing Operations may be abbreviated MFG. See Section 3.3.4.3 for a discussion of node references.

1. When there is text, it shall accompany the associated graphic diagram.

2. In non-publication models, the glossary associated with a specific graphic diagram shall accompany the diagram and shall define only the key words, phrases and acronyms used with the particular node.

3. In publication models, a glossary section shall define the key words, phrases and acronyms in alphabetical order for the entire model.

4. When a table of contents is provided for a model, it shall be presented as a node tree or node index, and shall contain node numbers, diagram titles and box names.

The desire of the U.S. Air Force to reduce costs and lead times by assisting the aerospace industry in its modernization efforts is evidenced in its many "Tech Mod" (Technology Modernization) programs. A similar goal, but using an industry-wide target rather than individual companies, was established under the ICAM (Integrated Computer Aided Manufacturing) Program. In ICAM, the goal was to develop "generic subsystems" which could be used by a large number of companies to provide a significant upgrade to the industry as a whole. These "subsystems" provide support for common functions such as management of information, shop floor scheduling and materials handling.

This ambitious goal needed a common "baseline" communication vehicle around which to plan, develop and implement the subsystems in the individual aerospace companies. The baseline was called the "Architecture of Manufacturing", since it was to provide an industry-wide "architecture" showing how industry works today and around which generic subsystems could be planned, developed and implemented.

To develop the architecture, a "language" was needed in which to express and document current aerospace industry operations. At the outset of ICAM, the Air Force issued a Request for Proposal to build the architecture. An activity modeling technique was specified as the expressive language (where an activity was defined as a manufacturing cell or operational unit). To be successful, the language had to satisfy the following criteria:

- Since the architecture was to depict manufacturing, it had to be able to express manufacturing operations in a natural and straightforward way.

- Since the subject was so vast and complex, it had to be concise and provide a straightforward means of locating details of interest easily and quickly.

- Since it was to be used by a wide audience, it had to be able to communicate to a wide variety of aerospace industry personnel as well as to Air Force ICAM Program Office personnel.

- Since it was to serve as a baseline for generic subsystem planning, development and implementation, it had to permit sufficient rigor and precision to insure orderly and correct results.

- Since the baseline was to be developed through the cooperative effort of a large segment of the aerospace industry, it had to include a methodology (rules and procedures) for its use that would permit many diverse groups to develop architecture pieces and that would permit wide-spread review, critique and approval.

- Since the baseline was to represent the entire aerospace industry rather than any one company or industry segment, the technique had to include a means of

separating “organization” from “function”; that is, a common agreement could not be achieved unless the individual companies’ organizational differences were separated out and only the common functional thread was captured.

The SADT™ (Structured Analysis and Design Technique™) originally developed in 1972 by Douglas T. Ross, of SofTech, was selected as the “The Architecture Method” for use in the Air Force Computer Aided Manufacturing (AFCAM) Project. The activity modeling technique was further developed and used in the follow-on ICAM Part I Program. The major subset of this technique used by the ICAM Part II Program Office was later re-named and documented as “IDEF0”.

The original IDEF0 methodology incorporated basic concepts which address each of the needs listed above. These basic IDEF0 concepts are:

1. Activity Modeling Graphic Representation. The “box and arrow” graphics of an IDEF0 diagram show the manufacturing operation as the box, and the interfaces to/from the operation as the arrows entering/leaving the box. In order to be able to express real-life manufacturing operations, boxes may be interpreted as operating with other boxes, with the interface arrows providing “constraints” as to when and how operations are triggered and controlled.

2. Conciseness. The documentation of a manufacturing architecture must be concise to permit encompassing the subject matter. The linear, verbose characteristic of ordinary language text is clearly insufficient. The two-dimensional form provided by a blueprint-like language has the desired conciseness without losing the ability to express relationships such as interfaces, feedback and error paths.

3. Communication. There are several IDEF0 concepts which are designed to enhance communications:

   - Diagrams based upon very simple box and arrow graphics.
   - English text to specify box (function) and arrow (data or objects) meanings.
   - Gradual exposition of detail, featuring a hierarchy with major functions at the top and successive levels of sub-functions revealing well-bounded detail breakout.
   - A node index for locating details within the hierarchic structure of diagrams.
   - Limitation of detail on each successive diagram to not more than six sub-functions for ease of reader comprehension.
   - Diagrams supported with text and glossary to increase the preciseness of the graphic representation.

4. Rigor and Precision. The rules of IDEF0 enforce sufficient rigor and precision to satisfy ICAM architecture needs without overly constraining the analyst. IDEF0 rules include:

- Detail exposition control at each level (3-6 box rule).
- Bounded context (no omissions or additional out-of-scope detail).
- Syntax rules for graphics (boxes and arrows).
- Uniqueness of names and labels on a diagram.
- Diagram connectivity (Detail Reference Expressions [DRE]).
- Data/object connectivity (ICOM codes and tunneled arrows).
- Input vs. control separation (rule for determining role of data or objects).
- Minimum control of function (all functions require at least one control).
- Arrow branch (fork or join) constraint (labels for arrow segments).
- Arrow label requirements (minimum labeling rules).
- Purpose and viewpoint (all models shall have a purpose and viewpoint statement).

5. Methodology. Step-by-step procedures are provided for modeling, review and interview tasks.

6. Organization vs. Function. The separation of organization from function is included in the purpose of the model and carried out by the selection of functions and arrow labels during model development. Continual review during model development ensures that organizational viewpoints are avoided.

In the remaining sub-sections descriptions of some of the basic concepts are elaborated to clarify them and show their utility.

The IDEF0 methodology may be used to model a wide variety of automated and non-automated "systems" or subject areas, including any combination of hardware, software, machines, processes or people. For new systems IDEF0 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 is a model. A model consists of diagrams, text and glossary, cross-referenced to each other. Diagrams are the major components of a model. All functions and interfaces are represented as boxes (functions) and arrows (data or object interfaces) on diagrams.

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. Call arrows, a type of mechanism arrow which enables the sharing of detail between models or between portions of the same model, connect to the bottom of the box and point downward. These arrow positions are illustrated in

Figure A1.

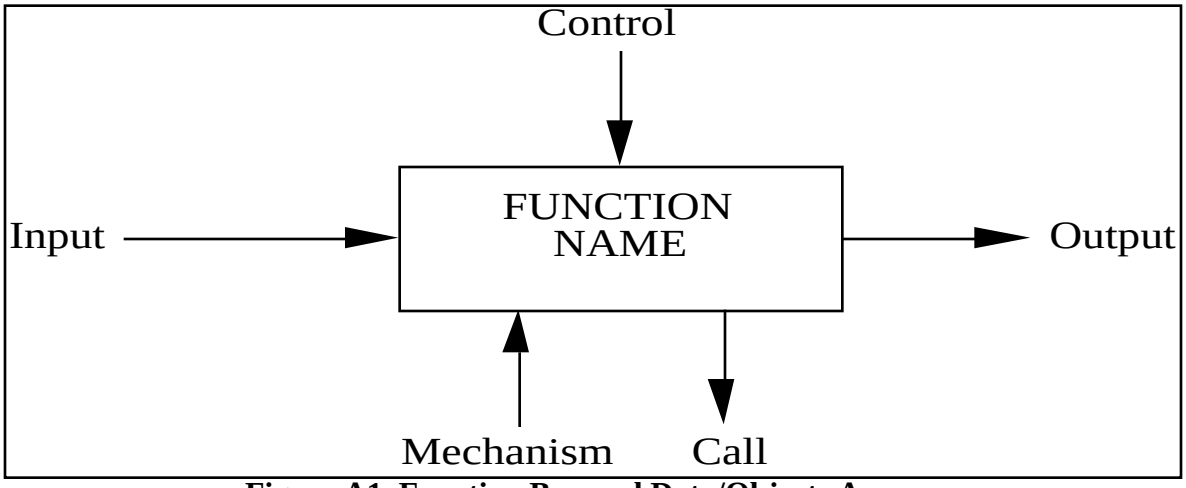


Figure A1. Function Box and Data/Objects Arrows

These box and arrow meanings are used to relate several sub-functions on a diagram comprising a more general function. This diagram is a "constraint diagram" which shows the specific interfaces which constrain each sub-function, as well as the sources and targets of the interface constraints (Figure A2).

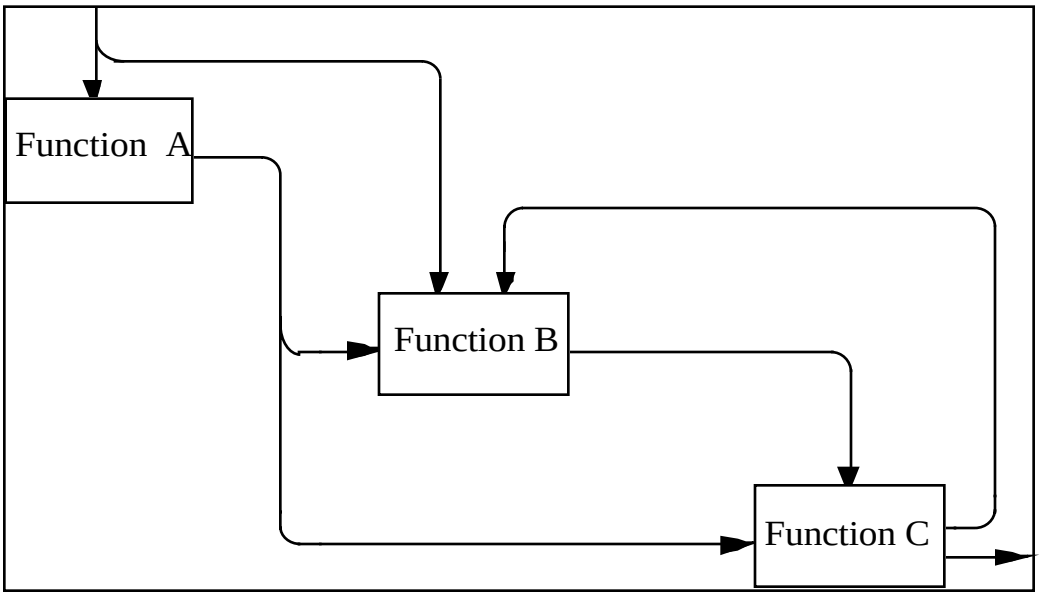


Figure A2. Constraint Diagrams

In Figure A2, Function B is constrained by one input and two controls, and produces a single output, which constrains Function C.

Here, the term "constrains" means that a function uses the data or objects shown entering

the box, and therefore, is constrained from operating by the interface; the function cannot act until the contents of the interface arrow are provided, and the way in which the function operates depends upon the details (value, number, etc.) of the contents of the interface arrow.

One of the most important features of IDEF0 is that it gradually introduces greater and greater levels of detail through the diagram structure comprising the model. In this way, communication is enhanced by providing the reader with a well-bounded topic with a manageable amount of new information to learn from each diagram.

The structure of an IDEF0 model is shown in Figure A3. Here, a series of four diagrams is shown with each diagram's relation to the others.

An IDEF0 model starts by representing the whole system as a single unit - a box with arrow interfaces to functions outside the system. Since the single box represents the system or subject area as a whole, the descriptive name written in the box is general. The same is true of the interface arrows since they also represent the complete set of external interfaces to the system as a whole. The diagram with the single box is called the “context diagram,” and shall define in text the viewpoint and purpose of the model.

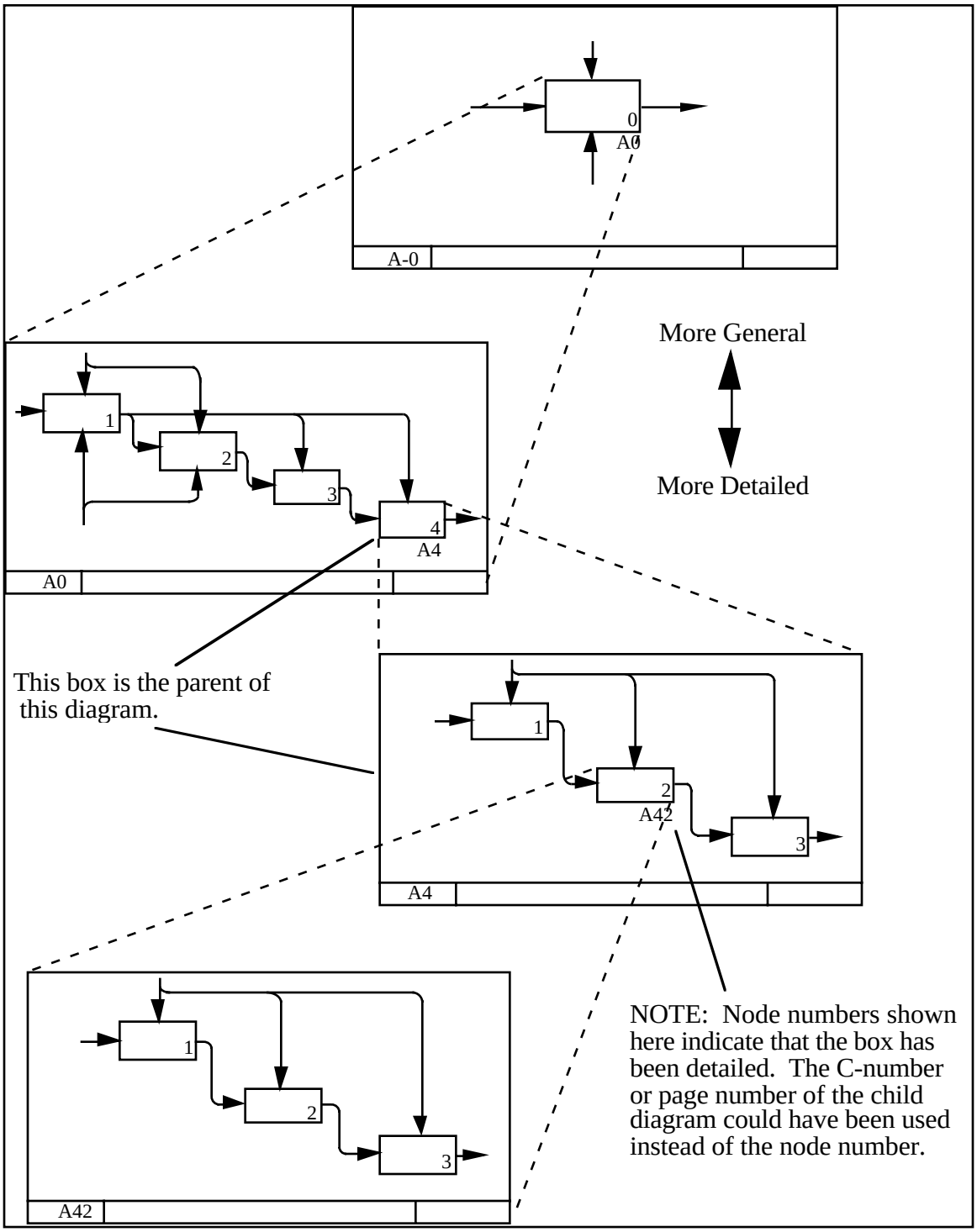


Figure A3. IDEF0 Model Structure

The box that represents the system as a single module is then detailed on another diagram with boxes connected by interface arrows. These boxes represent major sub-functions of the single parent function. This decomposition reveals a complete set of sub-functions, each represented as a box whose boundaries are defined by the interface arrows. Each of these sub-functions may be similarly decomposed to expose even more detail. In IDEF0, we use the following terminology: functions are “decomposed”; the boxes that represent functions are “detailed”.

A box, if detailed, is always detailed on a child diagram into no fewer than three boxes, but no more than six boxes. The upper limit of six forces the use of a hierarchy to describe complex subjects. The lower limit of three insures that enough detail is introduced to make the decomposition (detailing) of interest.

Each diagram in a model is shown in precise relationship to other diagrams by means of interconnecting arrows. When a function is decomposed into sub-functions, the interfaces between the sub-functions are shown as arrows. The name of each sub-function box plus its labeled interfaces define a bounded context for that sub-function.

In all cases, every sub-function is restricted to contain only those elements that lie within the scope of its parent function. Sub-functions are discrete and do not overlap. Further, the collection of sub-functions cannot omit any elements. Thus, as already indicated, the parent box and its interfaces provide a context for its child diagram. Except for tunneled arrows, nothing may be added or removed from this precise boundary.

The IDEF0 methodology includes procedures for developing and critiquing models by a large group of people, as well as integrating support subsystems into an IDEF0 Architecture. Additional supporting procedures, such as librarian rules and procedures, and review cycle (see Annex C) procedures are also included in the IDEF0 methodology. (It should be noted that some of these rules and procedures, such as the Kit Cycle or Reader-Author Cycle critique procedures, are also used with other IDEF techniques.)

The creation of an IDEF0 model is the most basic of these “disciplined teamwork” procedures. The creation of a model is a dynamic process which usually requires the participation of more than one person. Throughout a project, authors create initial diagrams which are distributed to project members for review and comment. The discipline requires that each person expected to make comments about a diagram shall make them in writing and submit them to the author of the diagram. The author replies, also in writing. This cycle continues until the diagrams, and eventually the entire model, are officially accepted.

IDEF includes procedures for retaining written records of all decisions and alternate approaches as they unfold during the project. Copies of the diagrams created by an author are critiqued by knowledgeable readers who document suggestions directly onto the copies. Authors respond to each comment in writing on the same copy. Suggestions are accepted or rejected in writing along with the reasoning used. As changes and corrections are made, outdated versions of diagrams are retained in the project files.

The diagrams are changed to reflect corrections and comments. More detail is added to the model by the creation of more diagrams which also are reviewed and changed. The final model represents an agreement on a representation of the system or subject area from a given viewpoint and for a given purpose. This representation can be easily read by others outside the initial project, used for presenting the system definition in short stand-up briefings or in walk-throughs, and for organizing new projects to work on system changes.

A model is made up of a collection of diagrams and associated materials arranged in a hierarchic manner. A node index (or table of contents) shall be provided. Placing the diagrams in hierarchical order gives an overall view of the model and allows access to any portion.

Reading is done top-down**,** considering each diagram as a context bounded by its parent box. After the top level diagrams are read, first level diagrams are read, then second level diagrams are read, etc. If specific details about a model are needed, the node index is used to descend through the levels to the required diagram.

When published, a model is bound in "page-pair" format and "node index" order. "Page-pair" format means that each diagram and the entire text associated with it appear on a pair of facing pages. (Figure B1.)

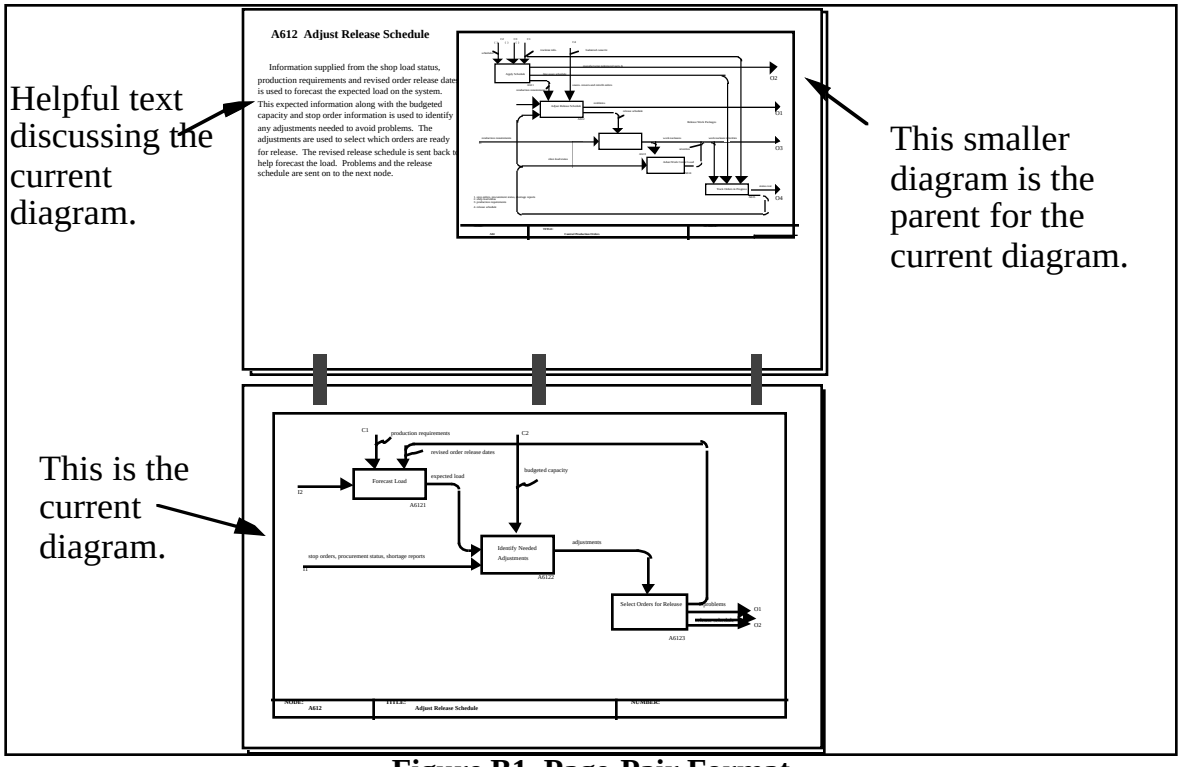


Figure B1. Page-Pair Format

"Node index" order means that all child diagrams relating to one box on a diagram are presented before the children of the next box. This places related diagrams together in the same order used in an ordinary table of contents. (Figure B2.)

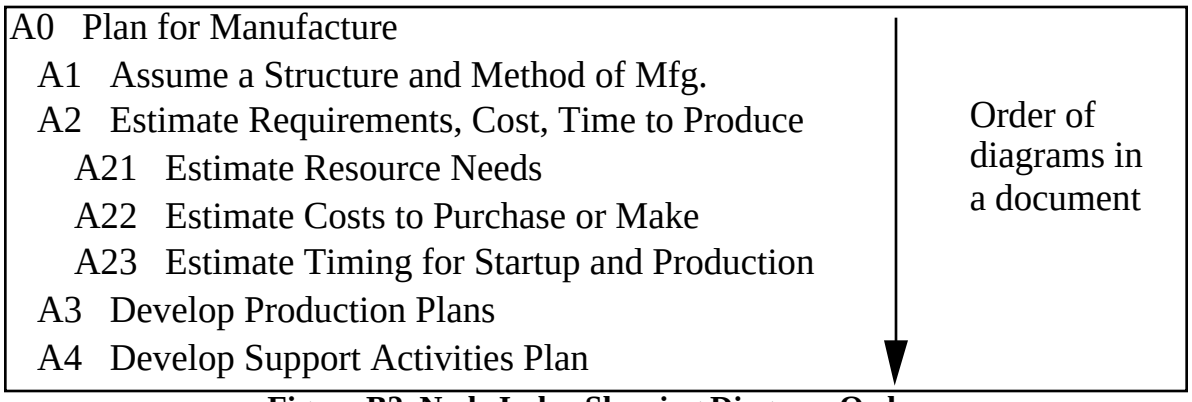


Figure B2. Node Index Showing Diagram Order

Models provide an overview of the whole system or subject area as well as details of a particular subject. To read a model for its overview, use the index to find all high-level diagrams. (Figure B3.)

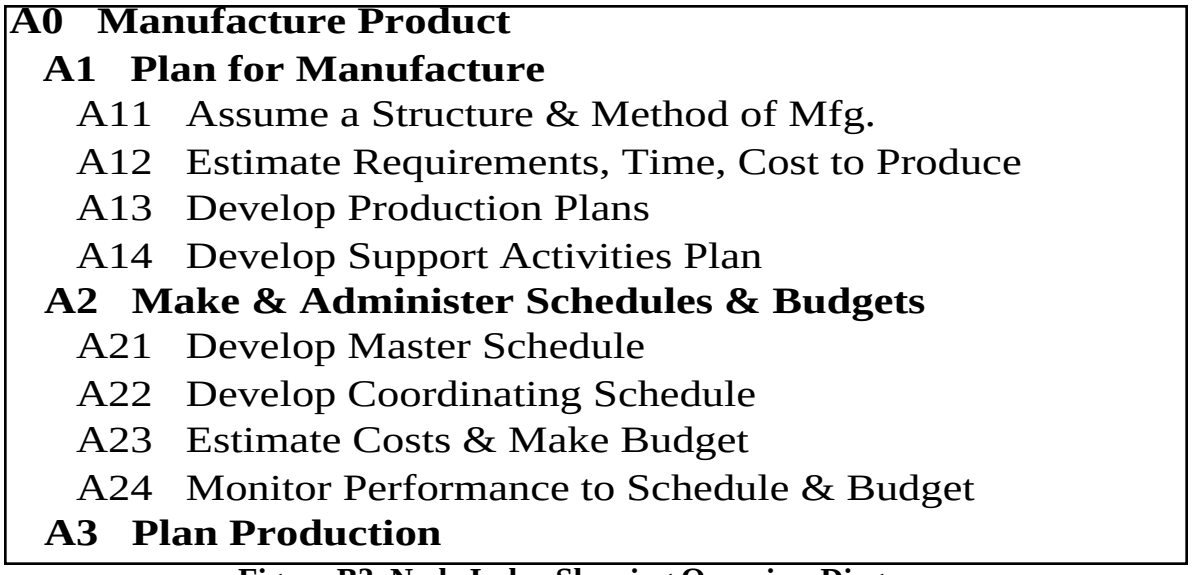


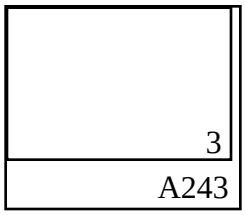
Figure B3. Node Index Showing Overview Diagrams

To read a model for detail, use the index to find all diagrams detailing the subject of interest. (Figure B4.)

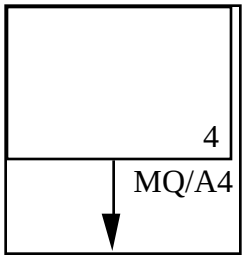
A0 Manufacture Product
A1 Plan for Manufacture
A11 Assume a Structure & Method of Mfg.
A12 Estimate Requirements, Time, Cost to Produce
A13 Develop Production Plans
A14 Develop Support Activities Plan
A2 Make & Administer Schedules & Budgets
A21 Develop Master Schedule
A22 Develop Coordinating Schedule
A23 Estimate Costs & Make Budget
A24 Monitor Performance to Schedule & Budget
A3 Plan Production

Figure B4. Node Index Showing Specific Detailed Diagram

Further detailing in a model may be traced by referring to the detail reference expression (DRE) just below the box number. This indicates the node number, C-number or page number of the child diagram that details the box. In the example below, details for box 3 on diagram A24 may be found on a diagram with node number A243. If no DRE appears, the box has not yet been detailed.



Details may be shared within a model or between different models. In both cases, a call arrow (downward pointing) indicates where the shared detailing appears via a reference expression that may include a unique, abbreviated model mane. In the example below, box 4 is detailed by diagram A4 in model MQ. In this example, the reference expression is a diagram node reference.



The precise information about a system is in the diagrams themselves. The following reading sequence is recommended:

1. Scan the boxes of the diagram to gain an impression of what is being described.

2. Refer back to the parent diagram and note the arrow connections to the parent box. Try to identify a “most important” input, control and output.

3. Consider the arrows of the current diagram. Try to determine if there is a main path linking the “most important” input or control and the "most important" output.

4. Mentally walk through the diagram, from upper left to lower right, using the main path as a guide. Note how other arrows interact with each box. Determine if there are secondary paths. Check the story being told by the diagram by considering how familiar situations are handled.

5. Check to see if a related FEO diagram exists.

6. Finally, read the text and glossary, if provided.

This sequence ensures that the major features of each diagram receive attention. The text will call attention to anything that the author wishes to emphasize. The glossary will define the author's interpretation of the terminology used.

Each diagram has a central theme, running from the most important incoming boundary arrow to the most important outgoing boundary arrow. This main path through the boxes and arrows outlines the primary function of the diagram. (Figure B5.) Other parts of the diagram represent qualifying or alternative conditions which are secondary to the main path.

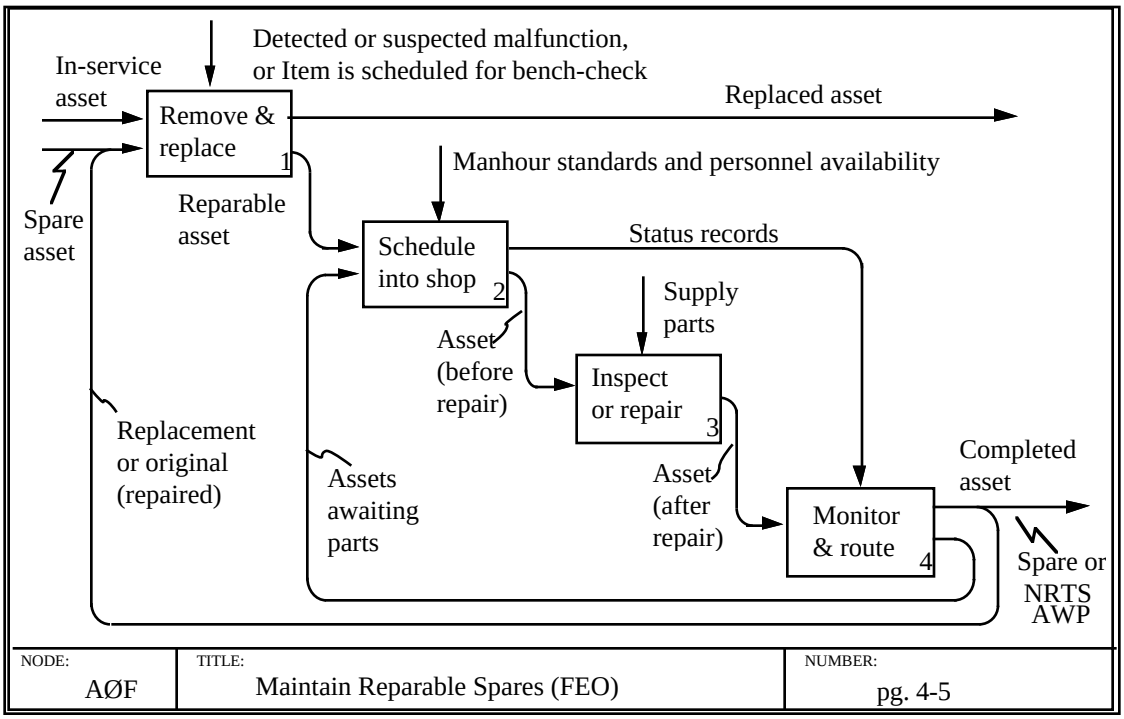


Figure B5. Example of Main Path

The system's operation can be mentally envisioned by pursuing the main path. Specific kinds of data inputs, the handling of errors and possible alternative outputs lend detail to the story. This walk-through enhances understanding of the diagrams.

The fundamental notion which must guide the interpretation of any diagram or set of diagrams is: Only that which is explicitly stated is necessarily implied. This derives from the very nature of constraint diagrams. Unspecified constraints must not be assumed; necessary constraints must be explicit. The corollary is that: Any further detailing not explicitly prohibited is implicitly allowed.

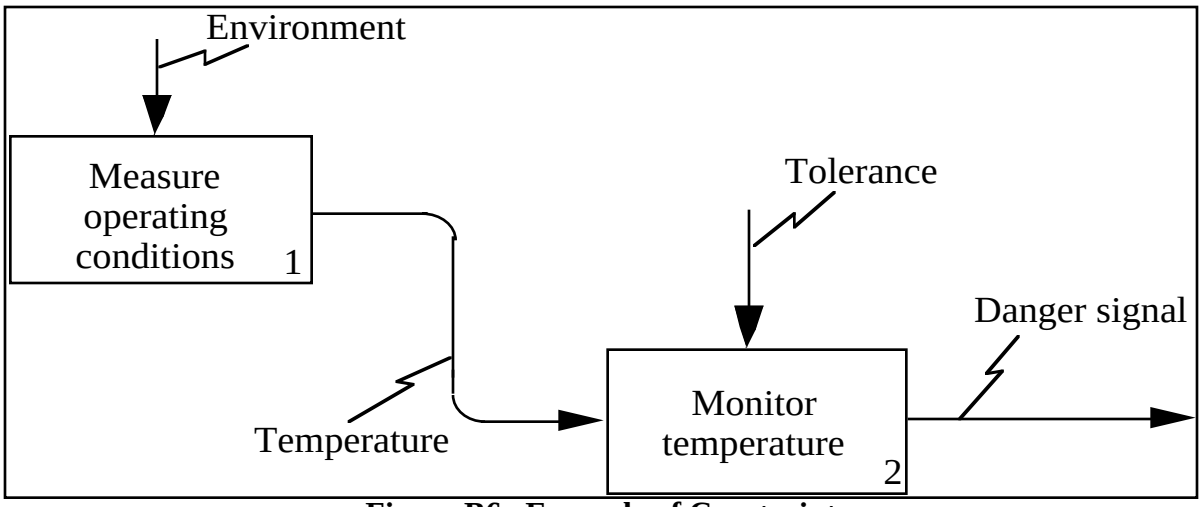


Figure B6. Example of Constraint

An assumption can be made using Figure B6 that the temperature is measured "often enough" and the tolerances are changed "when appropriate" and the temperature is monitored against the tolerances "often enough" that the danger signal will be produced "soon enough". None of these intuitive understandings would conflict with subsequent detailing which showed that:

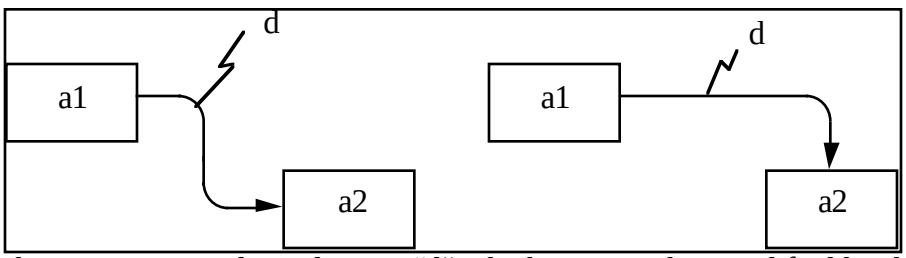
a.the temperature was measured by periodic sampling, or

b.current tolerances were requested only when the temperature increased by some fixed amount, or

c.a series of temperature values produced by box 1 was stored by box 2 which examined the pattern of change to determine if the pattern was within the tolerances, etc.

The graphic notations of a diagram are, by themselves, abstract. However, they do make important fundamental distinctions. Their abstract nature should not detract from the intended breadth of possible interpretations that are permitted.

Either of the two representations:



says that: the activity a2 is dependent on "d" which is created or modified by the activity

a1.

Each representation defines a constraint relationship between the two boxes. All that is explicitly stated by the intermediate arrow for either representation is expressed as follows: some activation of box 2 requires something called “d” that is produced by some activation of box 1.

Frequently, diagrams imply strongly that two or more boxes may need the contents of an arrow. The meaning of the boxes and arrows shown in Figure B7 is that something produced by box 1 is needed by box 2 and by box 3. It may be that an activation of the arrow’s “source” (box 1) must precede every activation of its “destination” (box 2 or box 3). It may be that one activation of the source is sufficient for every activation of any destination. Without additional information, the boxes and arrows alone permit either interpretation.

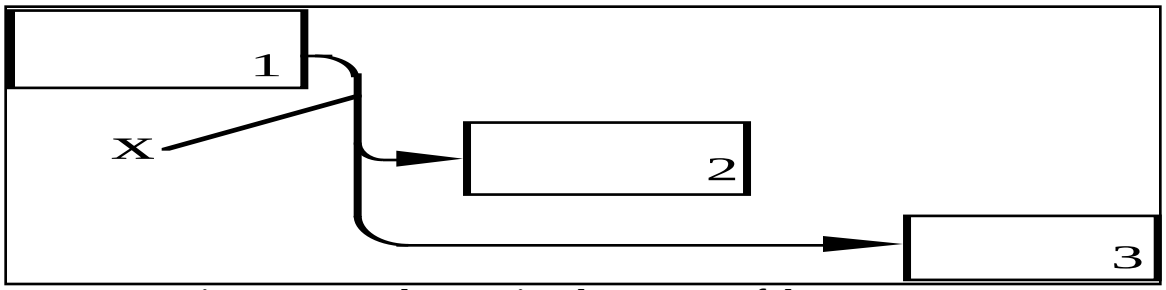


Figure B7. Two boxes using the contents of the same arrow

The basic interpretation of the box shown below (Figure B8) is: In order to produce any subset of the outputs [O1, O2, O3], any subset of the entries [I1, I2, I3, C1, C2, C3, C4, M1, M2, M3] may be required. In the absence of further detailing it cannot be assumed that:

a. any output can be produced without all entries present, or

b. any output requires all entries for its production.

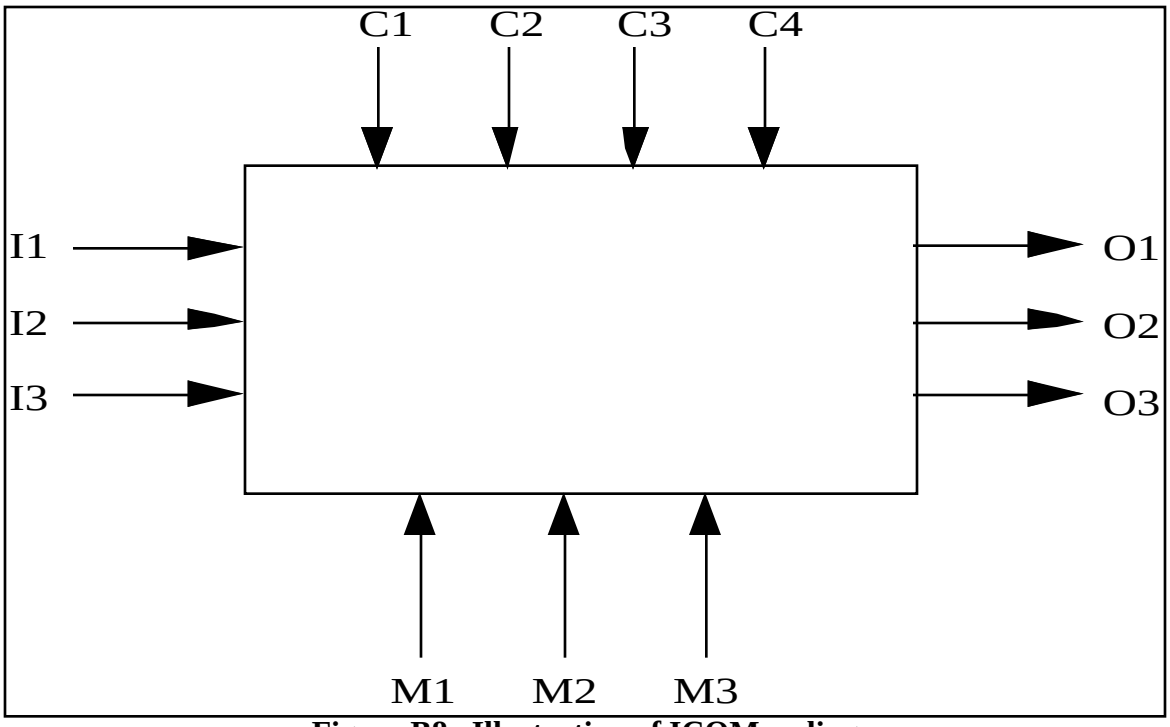


Figure B8. Illustration of ICOM coding

The partial detailing of the previous box (shown in Figure B9, as it might appear in a FEO diagram) indicates that I3, C2, C3 and C4 are not required for producing O1. Figure B9 illustrates the points that:

a. some form of further detailing will specify the exact relationship of inputs and controls to outputs;

b.until that detailing is provided, limiting assumptions about relationships “inside” each box should not be made;

c.reading of a diagram should concentrate on the arrows, which are explicit, rather than on box names, which are only implicit.

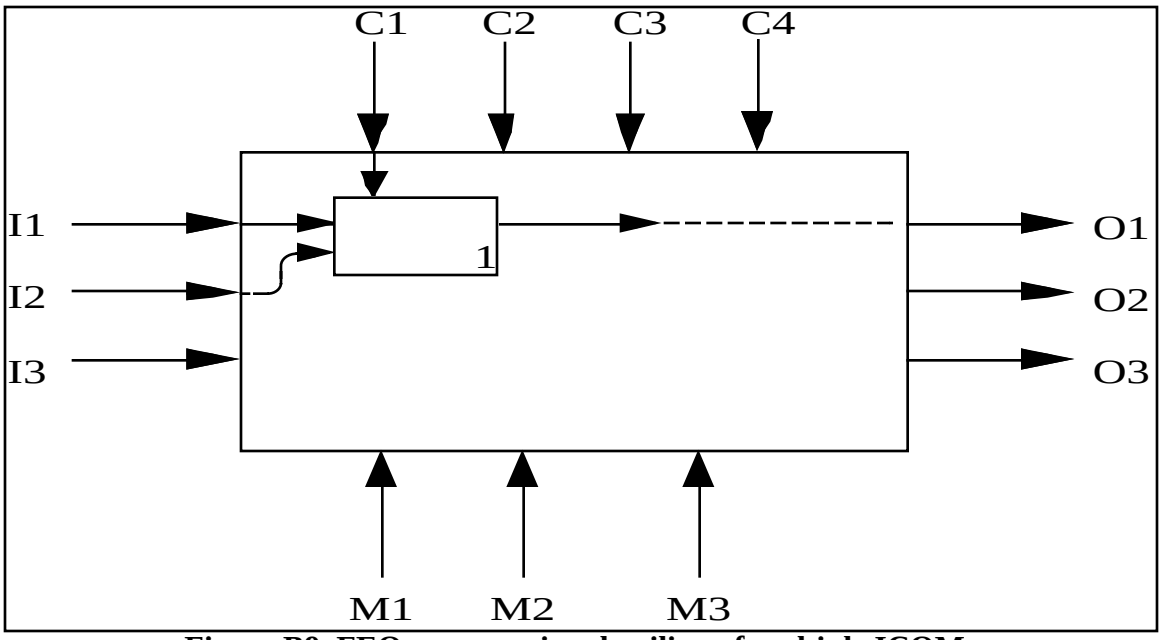


Figure B9. FEO representing detailing of multiple ICOMs

When creating any IDEF0 diagram, the requirements to be satisfied are that:

a.its purpose and viewpoint shall match the stated purpose and viewpoint of the overall model;

b.its boundary arrows shall correspond to the arrows that connect to its parent box;

c.its content shall be exactly everything in its parent box.

The step-by-step discipline of authoring makes it possible to create diagrams that form useful and coherent models. The discipline to follow is:

a.Bound the subject matter more precisely than the name of the function box suggests. This is done with a list of data and objects acted on or processed by the function.

b.Study the bounded set of subject matter and form possible sub-functions of the total function.

c. Look for natural patterns of connection of those sub-functions.

d.Split and cluster sub-functions to make suitable boxes.

e.Draw a final version of the diagram with careful attention to layout and clarity.

Before beginning any model, it is important to determine the model's orientation. This includes the context, viewpoint and purpose. These concepts guide and constrain the creation of a model. While they may be refined as authoring proceeds, they must be consistent throughout a model if its orientation is to remain clear and undistorted.

The context establishes the subject of the model as part of a larger whole. It creates a boundary with the environment by describing external interfaces. The context diagram establishes the context for the model.

The viewpoint determines what can be "seen" within the context, and from what "slant" or perspective. Depending on the purpose, different viewpoints may be adopted that emphasize different aspects of the subject. There is only one viewpoint per model.

The purpose establishes the intent of the model or the goal of communication that it serves. Purpose embodies the reason why the model is created (functional specification, implementation design, customer operations, etc.).

The starting point for every analysis is to bound the context. Decide what the focus is before the top-most box is created. Beware of drifting out of this carefully-selected starting domain. Every step should be checked against the starting purpose. Things that don't fit may be noted for later modeling of the relevant views. Clarity is derived from the rigors of detailing. Knowing how far to go, when to stop, when to change gears and how the pieces fit together will always depend on the purpose for which a model is created.

To start a model, create the A-0 diagram. Draw a single box containing the name of the function which encompasses the entire scope of the system being described. Use input, control and output arrows entering and leaving the box to represent the data and object interfaces of the system to its environment. This single-box diagram bounds the context for the entire model and forms the basis for further decomposition efforts. State the purpose and viewpoint on A-0 context diagram.

Some authors find it easier to sketch the A0 and then draw the single box and interface arrows shown at level A-0. It may be necessary to switch diagramming efforts back and forth between A-0 and A0 several times to obtain a good start for the decomposition.

If the A-0 diagram has begun at too low a level of detail, make the A-0 box the basis of a new level A0 diagram. Move up one level to a new A-0 and revisit the Viewpoint and Purpose statements. Repeat this process until an A-0 is reached which has sufficient scope to cover all aspects of the system. (Sometimes such a higher level will broaden rather than clarify the chosen viewpoint. If so, make an A-1 multi-box context diagram and keep the A0 diagram to the original intent.)

All system functions lie within the single box shown on the A-0 diagram. The diagram bounds the context of the system. The A0 diagram decomposes the single function on the A-0 diagram into its three to six major sub-functions.

The real "top" of the model is the A0 diagram. Its structure clearly shows what the A-0 diagram tried to say. The terms and structure of A0 also bound every subsequent level because it is a complete description of the chosen subject. Lower levels delineate the A0 functions (boxes). If the purpose of the model is to be achieved, this chain of detailing must be carefully followed at each step. Beginning at the top is the challenge of authoring. It forces the author to maintain a level of abstraction, keep an even model depth and relegate details to a lower level.

To form the structure of diagrams, detail each box on the A0 diagram into its three to six major parts. Form a new diagram for each box, which covers the same topic as its parent box but in more detail.

To detail each box by 3 to 6 child boxes, obtain the needed additional facts. Create a first-draft diagram by first listing all data and objects related to the function being decomposed. Take care that the list covers the entire topic of the parent box so that no portion is lost in the decomposition. Then draw boxes which associate candidate sub-function names with appropriate data and objects from the list, and draw arrows between the boxes.

To derive the clearest possible diagram, modify or re-draw the diagram several times until satisfied. Split (break up a box into two or more parts) and cluster (combine two or more parts into a single box) until satisfied. Split and cluster until you express the parent function in three to six boxes.

Generate portions of more detailed-level diagrams to explore points which need clarification. Create several (3 or 4) diagrams as a set, rather than one diagram at a time.

Eventually, each diagram will be accompanied by a page of narrative text, glossary and, perhaps, FEOs. The text associated with the A-0 diagram should complete the model's orientation and is written when the A-0 diagram is created. The text complements the context (expressed in A-0 itself) by expanding upon the stated viewpoint and purpose of the model.

Text for every other diagram (including A0) is quite different. It tells a brief, concise story. It does not duplicate what the diagram already says by merely describing each box function in words, but rather weaves through its patterns. At every level, this captures the viewpoint in a way that furthers the purpose. A graphic diagram may or may not have an associated text diagram.

The glossary explains the definitions the author gives to functions and data/objects in a diagram. These definitions are important because the terminology used in the model

may have a completely different meaning in one company from the meaning in another company. Terminology often differs among units or disciplines within the same company.

FEOs are diagrams that highlight a particularly interesting or subtle aspect of a diagram. They are not bound by IDEF box and arrow syntax and may contain partial arrow structures and notes to emphasize their point.

Given a complete parent diagram, “firm up” the higher levels before over-committing to details. That is, given A0, emphasize work on A1, A2, A3. Decomposing A1 into A11, A111, should be done later. This avoids potential rework should changes be made to higher level diagrams.

Keeping an even depth is not a strict rule. The amount of depth at any time depends on whether more depth would capture meaning better than one diagram. Don't put off doing a lower level diagram, e.g., A111, but sketch while the ideas are fresh. The important thing is to treat all such forays as sketches until the “horizontal” even level is confirmed. Be ready to rework the lower level material if it conflicts with higher level, e.g., A1, A2, A3.

Two guidelines are helpful in deciding which box to detail:

1. Start with the “hard part” - the part that is least familiar or is least clear.

2.Select the box whose detail will give the most information about other boxes.

The simpler topics can be more easily decomposed later, with less risk of error or oversight, and can be easily manipulated to fit the decomposition of the more complex issues.

Read Background: The author gathers information about the subject matter by reading source information.
Interview: The author personally interviews an expert on the subject matter.

Think: Digest the information obtained from reading and interviews before actual diagramming begins.

Pick Box: Decide which box is the appropriate one to detail based on information obtained.

Draw: This encompasses the actual creative process of generating a diagram. It is not limited only to drawing boxes and arrows. It also includes the listing of random data elements, making sketches, etc., which precede drawing boxes and arrows.

Re-draw: This covers the digestive stage of diagramming and corresponds to editing and reworking of verbal text. The activity here is concerned not with creating, but with graphical editing and rearranging for clarity.

Fix Master: This applies to the modification of master drawings to incorporate improve-

ments. It is primarily a mechanical operation, consolidating the results of separate re-drawings, often in response to reader comments.

Write and Edit Text:`Text accompanying any diagram should be precise. Editing will often alleviate unnecessary detail and redundancy.

Assemble: Assemble any material, diagrams, node trees, glossaries, text, etc., relevant to the subject. Include a completed Cover Sheet.

React: This refers to the author reacting to comments. It is a combination of reading and annotating reactions to the comments in response to readers.

Talk: This represents time spent when an author and reader actually get together and talk about author reactions to the comments.

Group Meetings: This is the time spent in group meetings reviewing progress or brainstorming next steps. The minutes of the group meetings will identify the subject matter under discussion.

Diagram creation is the most subjective and creative activity of the modeling process. It is open to wide variations between individual authors. No one set of steps will work equally well for all authors. The steps presented here are a proven sequence and are designed to assist a new author in drawing IDEF0 diagrams.

a. Create a relevant, but not yet structured list of data. List items within the context of the parent box that first come to mind. Group items, if possible, to show similarities.

b. Name functions that act on the listed data and draw boxes around the names.

c. Sketch appropriate arrows. As each box is drawn, leave arrow “stubs” to make the box more meaningful. Make complete connections as it becomes obvious what the diagram is saying.

d. Draft a layout that presents the clearest box and arrow arrangement. Bundle arrows together if the structure is too detailed. Leave only the essential elements, and modify diagram as necessary.

e. Create text, glossary and FEO diagrams, if necessary, to highlight aspects which are important. Propose changes, if needed, in the parent diagram.

Function boxes are generated using the major sub-functions of the parent. As sub-function names are written, draw boxes around them to form the start of an actual diagram. At this

stage the number of boxes is immaterial. They can be modified by clustering and splitting to conform to the three-to-six-box rule.

Clustering will group two or more boxes to form a single box. Its goal is to cluster related functions into a single, more general function. It eliminates premature detail which obscures the message to be conveyed at this level.

Splitting will break a single box into two or more parts. It is the inverse of clustering. Its goal is to provide more detail to present sufficient understanding of the subject being decomposed.

Review the resulting set of function boxes. Look for good balance between the factors chosen. See if the names can be made more specific. Use special terms and abbreviations only when needed to promote communication with the intended audience and only at the detailed diagram levels. Do not use them at the highest (A-0 and A0) levels. Carefully define special terms in the glossary.

In all cases, make the function box names verbs or verb phrases. Whenever the phrase can be interpreted as either a verb or a noun, use the notation "(v)" to signify the intended verb usage.

Boxes

1. In most cases, lay out boxes diagonally from upper left to lower right. While any layout which makes clear the author's intent is acceptable, vertical or horizontal formats tend to crowd arrows and hinder good structured analysis style.

2. Boxes placed in the upper left "dominate" boxes placed lower and to the right through the control arrows that link them. This standard style makes it easier for readers to understand your meaning.

3. Number each box in its lower right inside corner. Assign the box numbers on a diagram from left to right and from top to bottom. This will help define the node number for each box. The leading digits of the box's complete node number are the same as this diagram's node number. The last digit of the node number is this box number. If the box in Figure B10 appears in diagram A4, then the complete node number for this box would be A42.

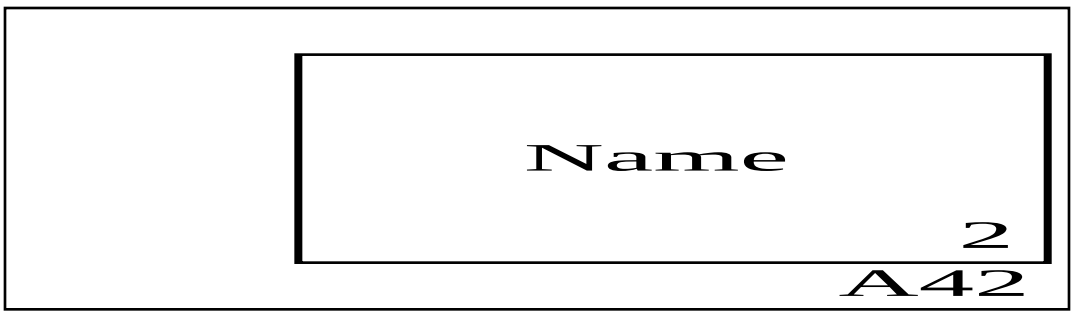


Figure B10. Box Number and Node Number

4. On working or draft copies, write the node number or C-number below the lower right corner of any box that is detailed. The C-number DRE indicates the specific version of the diagram that is intended.

5. No diagram may contain more than six boxes.

Sketch interface arrows connecting to each individual box. Connect ends of arrows to show which outputs supply which inputs and controls.

Recall that input data/objects are transformed by the function to produce the output. If an arrow contains both input and control data/objects, show it as a control. If it is uncertain whether an arrow is a control or an input, make it a control. If it is unclear whether or not a particular data or object arrow is needed at all, leave it out (as long as it is not the only control).

Output arrows show the results of possible activations of the function. The syntax for output arrows does not indicate which patterns of output arrows may occur under which circumstances. If the sequence is of particular interest, draw a FEO illustrating the pattern. Do not worry about sequence. Just make sure that all important cases are allowed by the diagram.

Bundle groups of related arrows whenever possible. The most common mistake when creating arrows is to make the arrow structure or the arrow labels too detailed. The level of detail of arrows should match the level of detail of boxes. At high levels, both box names and arrow labels will be general.

As a final check, compare all arrows to the data list to insure that each correct data item appears. Elements that do not appear either are inappropriate for this level of detail or were overlooked when creating the arrows.

Think control and constraint, not flow.

A basic rule for layout of the arrow structure is "constrain, don't sequence." That is, make the diagram structure show relationships that must be true no matter which sequence is followed.

Even though something must progress from stage to stage to reach some desired end result, try to express the constraints that must be satisfied or the invariant properties that must be true rather than some one specific sequence of steps that will yield that result.

The reason is that all boxes may be active simultaneously. Thus, sequence has no meaning.

It is always more powerful to constrain than to sequence. Wherever possible, diagrams should be created that say the right thing regardless of what steps are taken first. Clearly,

this is better than restricting to only one of the possible sequences.

Often, it is easiest at first to think of sub-function actions in a particular sequence to get unstuck and get something on paper. This may be a good way to get going, but always rework the first attempt into a constraint structure.

Label carefully.

Subordination of unneeded details highlights the meaningful ones. Don't clutter your diagrams with too much information and too many arrows. Don't spend too long on a single level. Everything need not be expressed at once to avoid being incomplete and misunderstood. The whole point of the discipline is to get everything said, eventually.

Also, if there is too much in a diagram, it becomes rigid. If this is allowed to happen, the diagram is weakened. Strength comes from the structure. This can be accomplished by leaving details to the sub-functions. Do iterate between high-level diagrams and sub-functions that fill out the details.

Leave out questionable arrows.

It is often hard to determine whether to show an arrow or not. The easiest way to handle the arrow question, is "When in doubt, leave it out." If the arrow isn't really essential to the main backbone, if there are questions about it, it is probably incorrect to include it.

Incorrect omission of a questionable arrow now won't cause permanent damage. The need for the arrow will be clearer in considering sub-functions and the ICOM discipline will force the arrow back up to this level. At that time, there will be no question about it.

The initial goal in generating a diagram should be a clear diagram that represents a definite message and does not violate any syntax rules. When the diagram is finished, the critical guidelines of reading and review by others can be used to improve the first try. Most diagrams can be modified to make a second version that is in some sense better than the first. The first will rarely be the very best.

As skill levels develop, first diagrams get better and authors feel more comfortable using IDEF0. Reworking of diagrams will always be a necessary part of the process. The key idea is to use a review cycle to make progress on paper. In a series of orderly advances, all of the important aspects will be properly handled.

IDEF0 is a thought-forming methodology, not just a diagram-making exercise. Putting thoughts on paper and letting the notation and discipline work is a move towards a satisfactory resolution. Rely on an ability to ask good questions, rather than on the expectations of providing "perfect" answers.

When first creating a diagram, 3–6 function boxes of approximately the same level of detail are derived. Clustering and splitting will provide a boundary which is more easily understood or which will provide a simpler interaction between the function boxes.

Most often, clustering and splitting work together. Boxes are split and the resulting pieces clustered into new boxes which more closely convey the intended message. The same subject matter is covered, but pieces are grouped in a more understandable way.

Split and rephrase.

It is important that all of the boxes of a diagram have a consistent flavor. Changes elsewhere should not make an existing sub-function seem out of place. Split and rephrase to restore the balance.

Sometimes a box does not flow with the other boxes of the current diagram. Frequently the trouble is that other aspects have undergone change and clarification. That which earlier

was a good idea, now has the wrong slant or flavor.

Divide the offending box by splitting it into two or more pieces, one of which still contains the essence of the original idea, but more pertinently and concisely. Do expect to change the wording of the box (or boxes). With the separation, new ideas become clearer and mesh more closely with related boxes.

Cluster and replace.

A solid abstraction is both clearer and more powerful than premature detail. Cluster related boxes and replace by a single encompassing box.

Frequently a good level of abstraction can be made even better by clustering several boxes into a more general view and postponing the details to the next level down. Draw a line around the cluster and replace them all with one box, suitably named. The extra level is not an added complexity. It is a better representation because the structure has been shown more clearly.

This phenomenon arises often in conjunction with splitting and is one of the most powerful methods of explaining functions.

Both arrows and boxes should be at a corresponding level of abstraction in a diagram.

There are two basic ways to achieve this:

1. Bundle arrows with the same source and destination under a single more general label, and make one arrow.

2. Rename some boxes (using Split and Cluster) to better distribute the sub-functions and re-label resulting arrows.

It is seldom true that an excess of arrows indicates a mistake. It may be that they are both accurate and precise. But it is always true that an excess of arrows is bad if the message is obscured. The ability of readers to understand what is being said should guide the number of arrows used.

The detailed understanding revealed by creating a new diagram may uncover errors or oversights in the parent diagram. Parent modification is a natural and anticipated event. When creating the arrow structure, the rule is that “if there is any doubt whether an arrow is needed by a function box, leave it out - later detailing will demonstrate whether or not the arrow is really needed.” This is the point where such questions are resolved for a specific reason rather than through the former speculation.

Parent diagram changes may present various degrees of difficulty. If the change can be accommodated by a revision to the immediate parent only, this is simpler than a revision which involves more remote diagrams as well. When proposing a change, think it through carefully and assess its complexity. Substituting simple changes for major ones can harm the quality of the decomposition. When the correction is completed, check all boundary connections to insure that ICOM codes are properly shown. Inform other authors working on the diagram of the changes.

Always have in mind the parent diagram for the box being detailed. It will aid in the creation process. If the detailed diagram doesn't fit the context, either the current work or the context is wrong. Change the context or change the current work. They must match.

An important aspect of understanding diagrams is the ability to find and understand facts that are needed. Node numbers show the structure of the function decomposition (box detailing). The arrow network provides interface connections.

ICOM codes are written on all arrows having one end unconnected on the diagram. These boundary arrows connect the arrow network across diagrams. Each boundary arrow is labeled with an ICOM code to specify the connection of the arrow to the parent box.

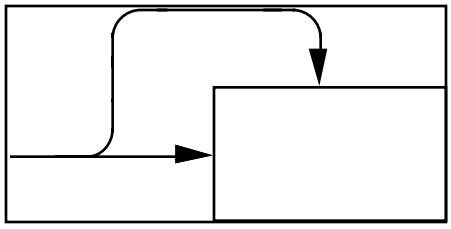
Lay out the boxes diagonally according to the constraint structure, from the upper left to lower right. Control feedbacks go up and left, and input feedback arrows (which model memory) go down and left. At this point, number the boxes from left to right.

It is best to start this layout with the most heavily used constraint arrows only, adding less used paths later. This subset of the arrows will permit the box position to be determined. Draw all boundary arrows shown on the parent diagram and then add the remaining arrows.

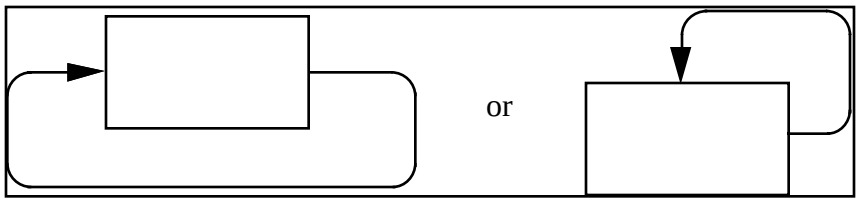
1. Function boxes shall always have control arrows, though they need not always have inputs.

2. When an entry arrow serves both control and input functions, show it as control. When in doubt, make it control. An arrow appearing on a parent box as control can appear on the next level as control or input or both, depending on its relationship to sub-functions at that level.

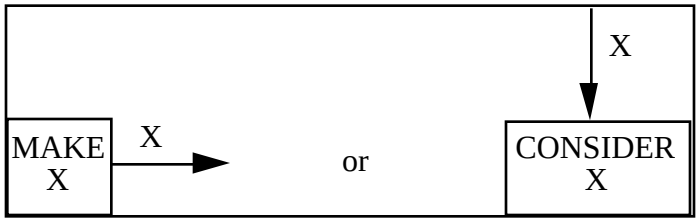
3. In general, do not split an arrow into both a control and an input to the same box. This detail is best shown on a lower level diagram where the destination of each branch and the reason for the split will appear. When it must be done to make the parent more meaningful, choose labels for the two branches that will convey your important decision.



4. Iterated activity (memory storage or feedback) may be shown as:



5. Try to avoid redundancies such as:



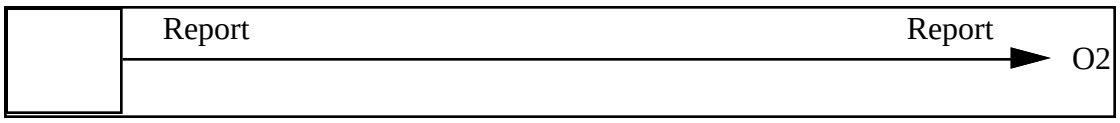
In these cases, the trivial box names merely repeat the message conveyed by placement of the arrows. A little additional thought will usually produce more informative box names.

1. Draw arrows along horizontal and vertical lines, not diagonally or as curves (except at corners).

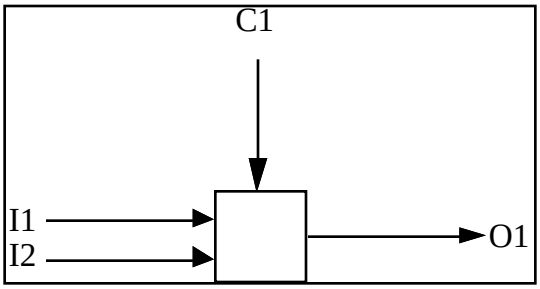
2. Place arrow corners, crossings and labels a reasonable distance away from boxes.

3. Don't use the keywords, i.e., "data", "function", "input", "output", "control"" or "mechanism" in names or labels, unless absolutely necessary.

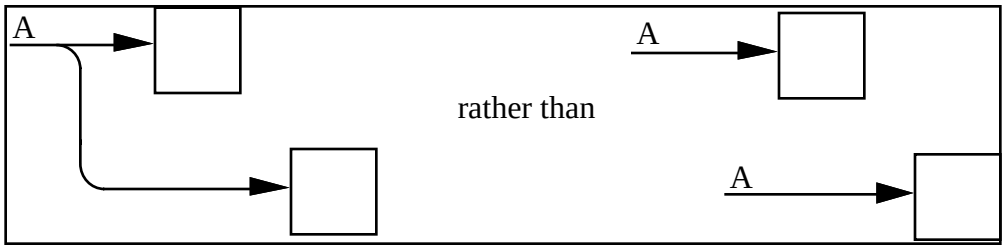
4. If an arrow is long, label it twice.



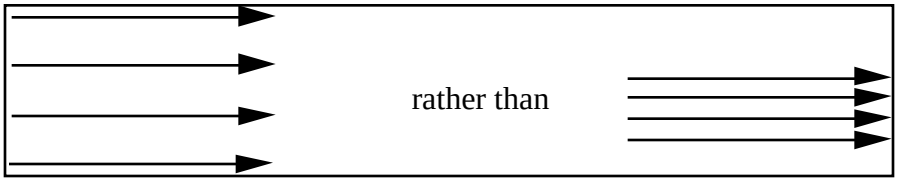
5. Place ICOM codes at the unconnected ends of arrows.



6. Connect open-ended boundary arrows to show all the places affected. Readers may miss connections otherwise.

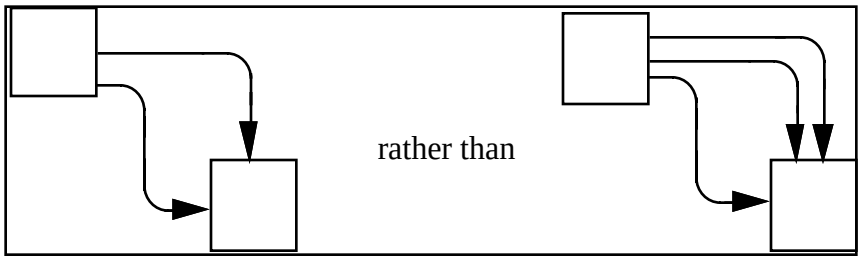


7. Space parallel arrows adequately. They are hard to follow visually if they are lengthy and close together.

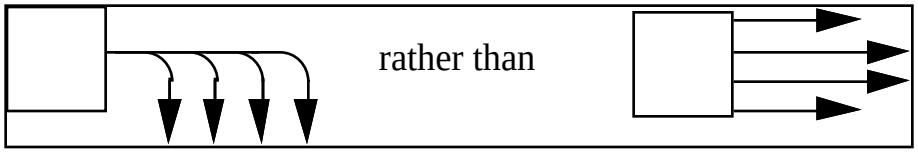


8. Place extra arrowheads along arrows where needed for clarity.

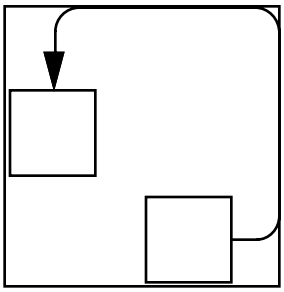
1. Bundle arrows with the same source and the same destination unless the arrow is of such importance that making it part of a pipeline would decrease clarity.



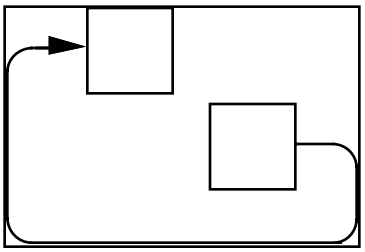
2. Use as few arrows as possible on any one side of a box. If there are too many, bundle some, label with a suitable abstract label and fan out branches to their destinations.



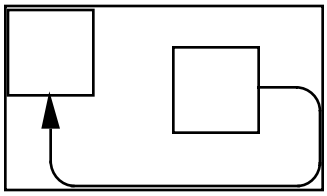
3. Control feedbacks shall be shown as “up and over.”



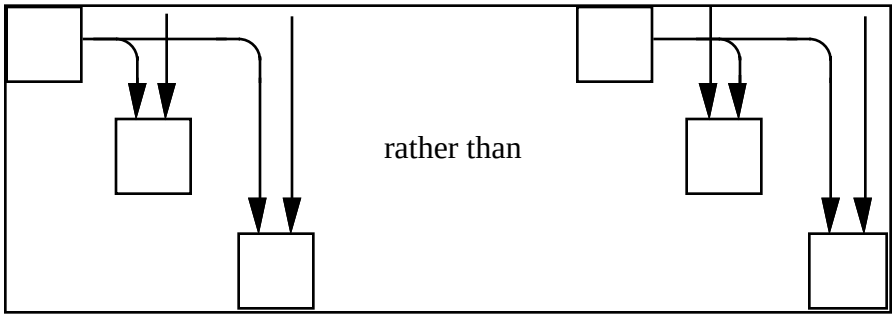
Input feedbacks shall be shown as “down and under.”



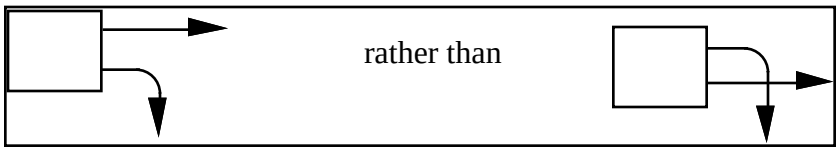
Mechanism feedback shall be shown as “down and under.”



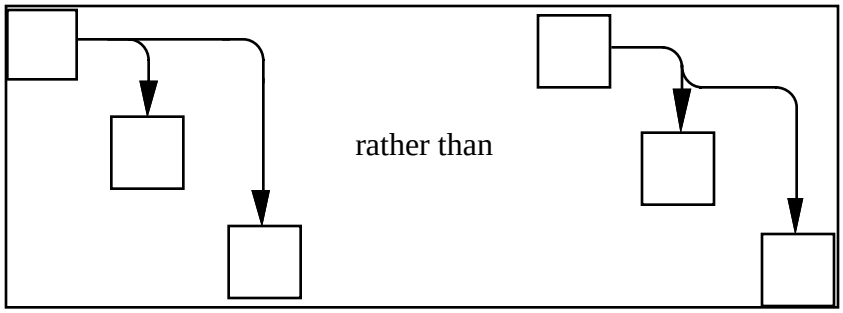
4. If an arrow forks and feeds into several boxes, draw it at the same relative ICOM position on each box, if possible.



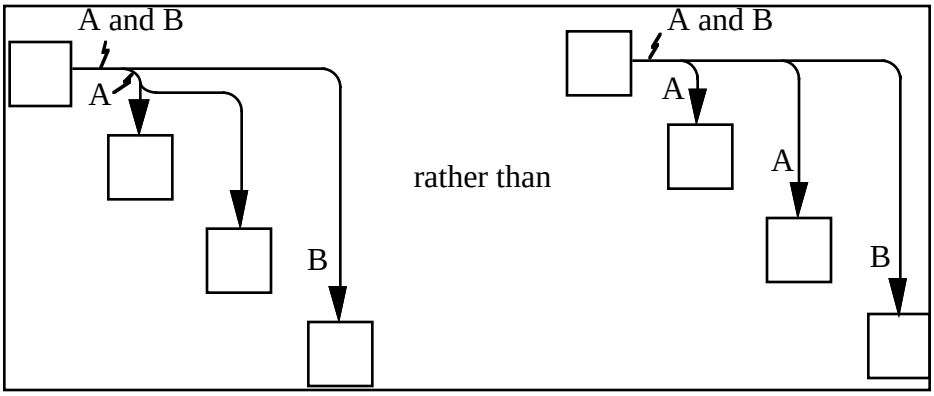
5. Lay out arrows so as to minimize crossings.



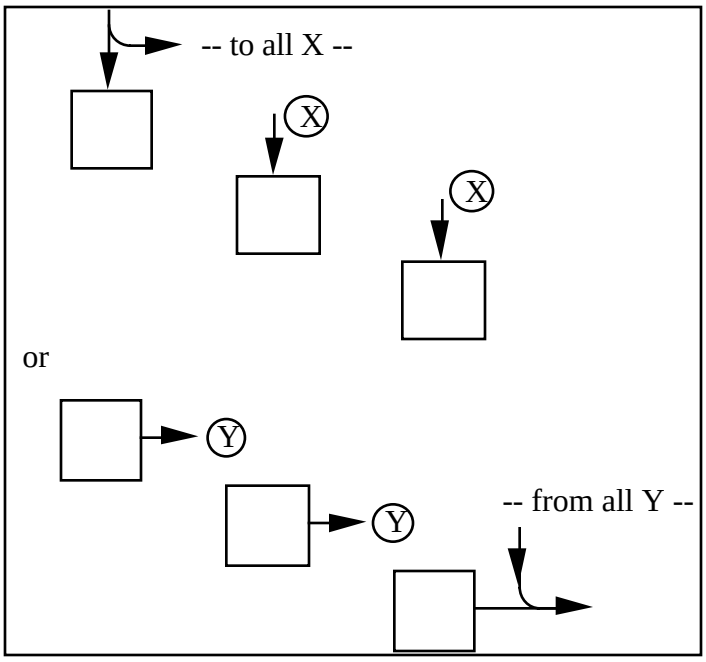
6. Minimize curves and corners, while keeping labeled segments clear:



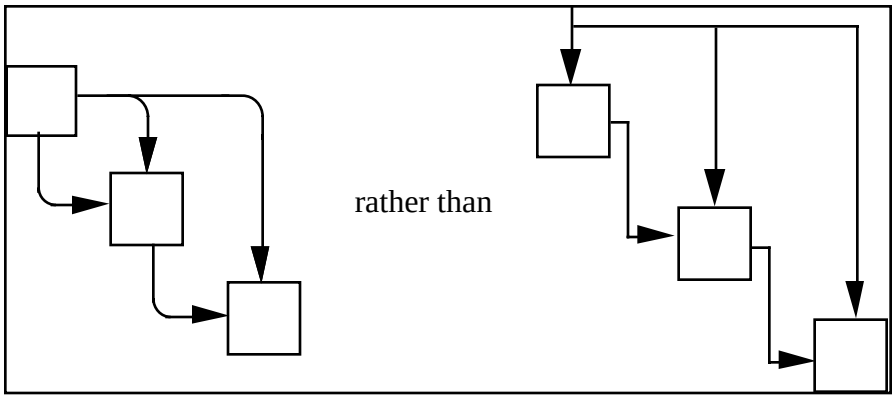
7. Use the expressive potential of branching arrows when and if it is appropriate.



8. To avoid clutter when showing an arrow which applies identically to or is obtained identically from every box on a diagram, use the “to all” convention:



9. All arrows shall have curved corners at joins, forks and bends.



The text that may accompany each diagram presents a brief integrating overview of the diagram, citing important relations, patterns, or subtle interactions between the boxes and arrows.

Preferably, the text is less than a page in length. It highlights features that the author feels are of special interest or significance by walking the reader through the main ideas of the diagram. It does not duplicate every detail shown on the diagram itself.

Write the text only after a diagram has received a fairly high level of review and approval. Waiting to write the text forces the diagram itself to properly communicate the intended

message. Text based on carefully drawn diagrams will be as structured and as organized as the diagram itself.

Use glossary definitions to summarize the special meanings that may arise for key terms, words, phrases and acronyms used in the diagram. A word or phrase may have different connotations for different readers of the diagram.

Try to get good text without adding a FEO. FEOs should be used to illustrate subtle aspects that clarify the intent of a diagram but which would clutter the diagram were they included. If a FEO is necessary, the text that accompanies it should refer to the related diagram.

There are two kinds of notes, model notes and reader notes. Model notes are discussed in the normative section.

In sharp contrast to the model notes, which are part of the diagram on which they appear, reader notes explicitly are on the diagram, not part of the diagram. Thus they cannot alter the meaning of the diagram's syntax or semantics. As with model notes, if the text of a reader note is to apply to several places or to several features of the diagram (including other model-notes or reader-notes) the circled-note-number (without any text) may be copied and may be attached by an IDEF0 squiggle to each point of application, but only on the single

diagram on which the reader-note's text appears.

By definition, reader notes are strictly informative, not normative. They may be about anything at all, with respect to the diagram or model. In practice, reader notes may be about the modeled subject matter or about its modeled presentation, including choice of words, layout, accuracy, etc.

Reader notes are denoted by a number "n" inside a small circle: . For any specific diagram

(and hence node number), the numbers, "n", shall form a consecutive sequence, starting at "1". All readers (including the author, when commenting about something) shall share the same reader-note sequence for a diagram. Reader-note numbers for a node shall never be reused or reassigned. Thus their creation sequence in the context of their node number is a permanent record of reader/author discussion regarding that node.

A standard notation is used in writing text and notes to refer to diagrams and specific parts of diagrams. References are based on box numbers, ICOM codes, node numbers, and note numbers.

For example,

O2	means	The boundary arrow with ICOM code O2
2I1	means	Box 2 Input 1

2O2 to 3C1	means	The arrow from 2O2 to 3C1
	means	Model Note n 
| n |	means	Model Note n (alternate notation)
(n)	means	Reader Note n (alternate notation)
	means	Reader Note n 
QA/A21.3C2	means	In "QA" model on diagram A21, see Box 3 Control 2
A42.	means	On diagram A42 in this model, see reader note 3 
A42.	means	On diagram A42 in this model, see model note 3 
A42.3	means	On diagram A42 in this model, see Box 3

These items may be used individually if they refer to the current diagram (e.g., in model notes or text). Otherwise they should be preceded by node number, and if necessary, by model name. A period "." is used to mean "see" a certain thing on the specified diagram. For example:

MFG/A21.3C2	means	In model "MFG", on diagram A21, see Box 3 Control 2.

Each reference needs only as many fields as are necessary to be completely unambiguous.

The fullest form is:

ACCT/(A21=BT56).1O2 to 4C3

which means in model "ACCT," diagram A21, C-number BT56, "see" the arrow from Box 1 Output 2 to Box 4 Control 3.

Embed neat and complete references in the wording of the text, glossary, and FEO pages, e.g., "The influence (2o3 to 1c2 and 3c2) of this cost on the ultimate price (o2) is of concern." This traces the third output of box 2, through boxes 1 and 3, to the O2 boundary arrow as the sentence is read. When finished with the text go through and add box-number references to tie the text to the diagram exactly.

Most of the time a simple box number ("Box 3") or a reference to a couple of arrows is suf-

ficient (“Box 3, O1 and C2”). When a reader needs to actually “see” the other diagram, use the “.” of the reference language.

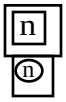
Just as ICOM coding naturally extends the node-number-based referencing scheme, the model-note and reader-note notations permit them to be referenced.

For example,

A21.3C2 |2| (5) reply

carries the previous example (“On diagram A21, see Box 3 Control 2”) to the case in which the reader references the diagram-author’s reply to reader-note #5 (perhaps added by the second reader of the diagram) which commented on the author’s model-note #2, regarding the control in question! In this example, “reply” shows the use of brief natural language expressions in the reference language.

This example also illustrates the use of abbreviations |n| for (model-notes) and (n) for



(reader-notes) for textual reference purposes. These abbreviations permit easier preparation of references to notes using standard word processors. References to notes in IDEF0 text and written correspondence are examples of textual references.

When analyzing or designing any system, it may be necessary to obtain or verify facts about the system or subject matter at hand. There are many sources of factual information.

One might do the following:

- Read existing documents, using each table of contents and index to locate needed information.
- Observe the system in operation, if it already exists.
- Survey a large group of people, through questionnaires or other such means.
- Talk to one or more experts who possess the desired knowledge.
- Use whatever is already known by the author.
- Guess or invent a hypothetical description, and ask readers to help bring it closer to reality.

Of all these methods, the most important is face-to-face interaction with an expert. Seldom will all existing information be written. Preconceived notions that are reflected in questionnaires are often faulty.

A key part of interviewing is to record the information obtained. This can be done either as informal notes, as activity and data/objects lists, as a formal matrix of functions, or as diagram sketches.

The purpose of an interview is to gather information from an individual who possesses an expertise considered important to the analytical effort. There are four types of interviews that might be conducted during the course of performing the analysis phase of an IDEF project.

(a)Fact Finding for understanding current operations. This type of interview is used to establish the content of a Current Operations Model or to help understand the existing environment.

(b)Problem Identification to assist in the establishment of future requirements. This type of interview is used to validate the Current Operations Model and to provide the foundation for a Future Operations Model.

(c)Solution Discussion regarding future system capabilities. This type of interview is used to establish the content of a Future Operations Model.

(d)IDEF Author/Reader Talk Session. This type of interview is used to resolve problems which have surfaced during the construction of an IDEF model.

The reason for identifying types of interviews is that during the course of performing an actual interview, ingredients of each type appear. The respondent might tell the interviewer facts about a given system in terms of problems. Also, the respondent might identify problems in terms of solutions to the problems. By constantly classifying the respondents' remarks, the interviewer can better appreciate the expert's point of view.

A "standard" interview kit can be used for recording the interview. It may be stored in an interview file and it may be distributed to appropriate project individuals. This distribution might include other members on the analysis team or even the interview respondent for corrections, additions and deletions. The interview kit would contain:

1.Cover Page (Kit cover)

2.Interview and Record Follow-up

(a)Interviewer Name (IDEF Author Name)
(b)Interview Date (IDEF Diagram Date)

(c)Interview Duration (Start time, End time)
(d)Respondent Name
(e)Respondent Title and Organizational Responsibility
(f)Respondent Telephone Number and Extension
(g)Additional Sources of Information Identified

(1)Documents—Title and Location
(2)Other Interviewees—Name, Title, Organizational Responsibility, Address, Telephone Number

(h)Essential Elements of Information—A Summary of the key points covered in the interview
(i)Follow-up questions and/or areas of concern either not covered during the interview or postponed
(j)New Terms for Project Glossary

3.Activity and Data/Objects List

4.Interview Agenda (developed in preparation for interview).

5.Interview Notes and Rough Diagrams

The development of any IDEF model is a dynamic process which requires the participation of more than one person. Throughout a project, the draft portions of a model are created by authors and distributed to others (project members, experts in the subject matter, management, etc.) for review and comment. These draft portions of a model are called Kits and may contain diagrams, text, glossary or any other information the author feels is pertinent to the development of the model.

It requires brief training and modest experience to correctly read and understand IDEF0 models. Such knowledgeable understanding is essential if the team-supplied quality assurance of IDEF modeling is to be realized. Everyone suitably advanced in correct reading skills is called a “reader”.

The IDEF teamwork discipline identifies all persons interested in the review of a model as reviewers. Reviewers who are assigned to make a written critique of a Kit are called commenters. Reviewers who receive a Kit for information only are not expected to make written comments and are called readers. The author and the commenters share responsibility for the quality of the model. Through their acceptance of the agreed result, the other reviewers share accountability for the utility of the results.

The discipline requires that each person expected to make comments about a Kit shall make them in writing using reader notes and submit them to the author of the Kit. Writing on the reader’s copy, the author responds to each note in writing (a simple check mark, for agreement; otherwise, a note in reply). This cycle continues, encompassing all Kits pertaining to a particular model, until the model is complete and recommended for publication.

At regular intervals during the evolution of a model, the master copy of the latest version is placed in the library, and a copy is disseminated in the form of a Kit, which is sent to readers to assist them in maintaining current information about the model. As the comments on each kit are reviewed by the author, he makes changes in the master copy of the model to incorporate corrections and changes. Another kit which includes the latest changes is then distributed to the list of readers. More detail is added by the creation of more diagrams, text and glossary. More comments are made; more changes are included. The end effect of this process for organized teamwork is a high assurance that the final IDEF models are valid, well expressed, and that a consensus has been reached by the set of readers who have been included in the review cycle.

In creating a document, materials written or gathered by an author are distributed, in the form of a Standard Kit, to commenters, reviewers and other readers. Commenters review the material and write comments about it. The commenters return the Kit to the author who reacts to comments and may use the comments to revise or expand the material. The Kit is returned to the commenter with the reactions from the author. Readers may return com-

ments to the author as well, but this is not required. This is known as a Kit Cycle (Figure C1).

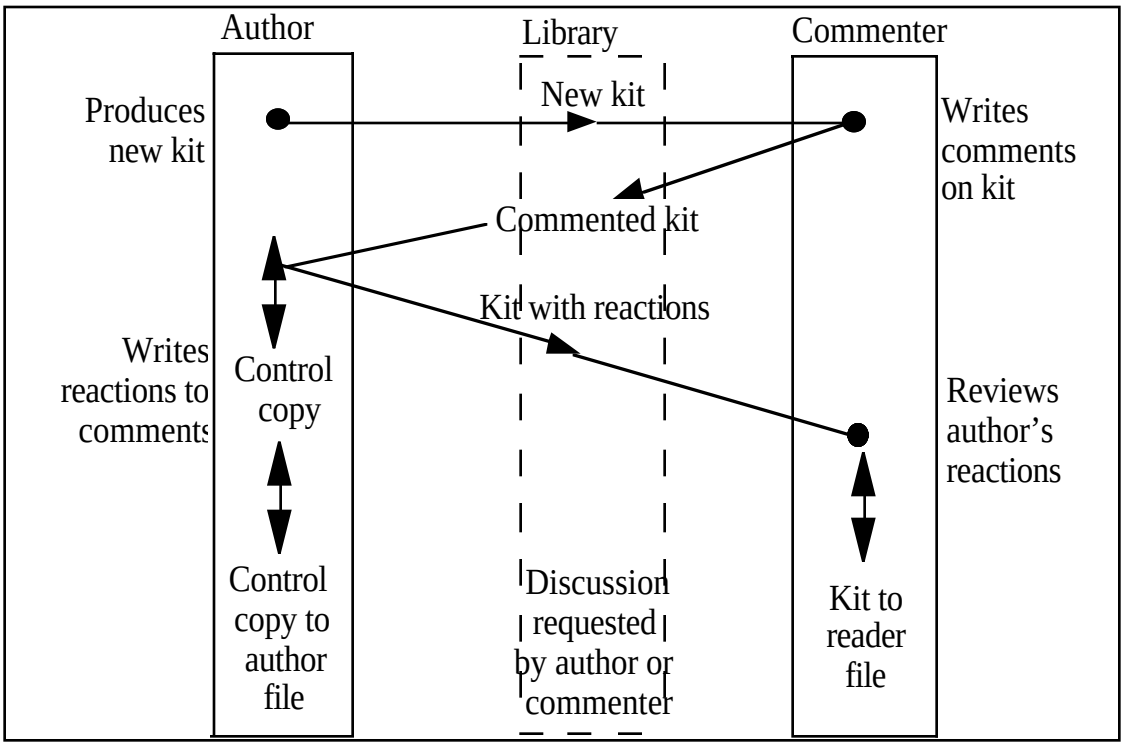


Figure C1. Kit Cycle

The steps of the Kit Cycle are as follows:

- The author assembles the material to be reviewed into a Standard Kit. A cover sheet is completed. Copies of the kit are distributed to each of the readers, and to the author. The original is filed for reference.

- Within the response time specified, the commenter reads the kit and writes comments directly on the copy in the form of reader notes, in red if possible. The kit is returned to the author.

- The author responds in writing directly on each commenter's copy, in blue if possible. The author may agree with the comment by check-marking it, noting it on his working copy and incorporating it into the next version of the model. If there is disagreement, the author writes a note of reply attached to the reader's note (no new note number). Whether or not there is disagreement, the kit is returned to the reader, completing one Reader/Author Cycle. (See C.4.1.2 regarding reader note numbering.)

- The reader reads the author's responses and, if satisfied, files the kit. (Commented kits are always retained by the reader.) If an assigned commenter does not agree with the author's responses, a meeting is arranged with the author to resolve dif-

ferences. If this cannot be done, a list of issues is taken to appropriate authority for decision. The author is not obligated to resolve differences with every reader, but commenters are disenfranchised if their concerns are not resolved.

This cycle continues until a document is created which represents the careful consideration of all project members. In addition, a complete history of the process has been retained.

The results of this Kit Cycle are a document to which author and commenters have contributed, and, if necessary, a list of issues that require management action.

Throughout the cycle, a project librarian handles copying, distribution, filing and transfer of kits between authors, commenters, reviewers and readers.

The roles and functions of people involved are:

- Authors (Analysts)
  People who prepare any IDEF model.

- Commenters (Experts or other Authors)
  People knowledgeable of the subject being modeled from whom authors may have obtained information by means of interviews, and have enough training in an IDEF technique to offer structured comments in writing. People assigned to make a written critique of a kit.

- Readers (Experts or anyone who wishes to be on the reader list)
  People knowledgeable of the subject being modeled from whom authors may have obtained information by means of interviews, and review documents for information but are not expected to make written comments.

- Librarian
  A person assigned the responsibility of maintaining a file of documents, making copies, distributing kits and keeping records.

All people playing these roles must be trained and experienced IDEF0 readers, so they can perform the assigned functions reliably.

A “role” has nothing to do with someone’s job title, and the same person may be asked to perform several roles. Thus, each individual’s participation is, in fact, unique and depends upon the kit involved.

An author interviews experts, analyzes the information, organizes it into diagrams and creates models. An author may or may not be the source of the technical content of a document. An author may serve only as an analyst of acquired information, identifying, sorting

out and organizing the presentation of knowledge obtained from experts and applying modeling skills in expressing his understanding in IDEF0 terms.

Assigned commenters read material produced by an author and verify its technical accuracy. They are responsible for finding errors and suggesting improvements. The role of a commenter is the key to producing high quality results. But every reader should note whether the author has followed the IDEF technique consistently; whether the viewpoint and purpose have been adhered to; and whether errors or oversights exist which should be brought to the author's attention. If a reader is a trained author, suggesting changes in the hierarchical breakdown, variations in activity box content and other observations to enhance the communication power and utility of the model will be welcomed by most authors.

In this section, general guidelines for readers and authors are discussed—readers may have further special interests as commenters and reviewers that are not covered here.

No set pattern of questions and rules can be adequate for commenting, since subject matter, style and technique vary so widely. However, guidelines do exist for improving quality. The major criteria for quality are: Will the document communicate well to its intended audience? Does it accomplish its purpose? Is it factually correct and accurate, given the bounded context? Overall guidelines for commenting are:

- Make notes brief, thorough and specific. As long as the author understands that niceties are dropped for conciseness, this makes for easier communication and less clutter.

- Use the notation (reader notes) to identify comments. To write an -note, 

check the next number off the READER NOTES list, number the note, circle the number and connect the note to the appropriate part with a squiggle  . (See Section C.4 Standard Diagram Form.)

- Make constructive criticisms. Try to suggest solutions or point out sources of problems, clearly.

- Take time to gather overall comments. These may be placed on the cover or on a separate sheet. (But don't gather specific points onto this sheet when they belong on the individual pages.) Agenda items for author/commenter meetings may be summarized. Make agenda references specific.

The length of time spent critiquing depends on a variety of things: familiarity with what is being described, the number of times something has been reviewed, the experience of the reader and author, etc. A kit returned to an author with no comments other than the reader's signature and a check mark means that the reader is in total agreement with the author. The reader should realize that there is a shared responsibility with the author for the quality of the work.

When a reader returns a kit, the author responds by putting a "√" or "X" by each -note (read-

er note). "√" means the author agrees with the commenter and will incorporate the comment into the next version of the kit. "X" means the author disagrees. The author must state why in writing where the comment appears. After the author has responded to all comments, the kit is returned for the reader to retain.

Until comments and reactions are on paper, readers and authors are discouraged from conversing.

When a meeting is required, the procedure is as follows:

1. Each meeting should be limited in length.

2. Each session must start with a specific agenda of topics which relate to one or more of the comments and author responses, and the session must stick to these topics.

3. Each session should terminate when the participants agree that the level of productivity has dropped and individual efforts would be more rewarding.

4. Each session must end with an agreed list of action items which may include the scheduling of follow-up sessions with specified agendas.

5. In each session, a "scribe" should be designated to take minutes and note actions, decisions and topics.

6. Serious unresolved differences should be handled professionally, by documenting both sides of the picture.

The result of the meeting should be a written resolution of the issues or a list of issues to be settled by appropriate managerial decision. Resolution can take the form of more study by any of the participants.

A Kit is a technical document. It may contain diagrams, text, glossaries, decision summaries, background information or anything packaged for review and comment.

An appropriate cover sheet distinguishes the material as a kit. The cover sheet has fields for author, date, project, document number, title, status and notes.

There are two types of IDEF Kits:

Standard KitTo be distributed for comment. It is considered a “working paper” to assist the author in refining his total model.

Update Kit Contains the latest version of a model. It is sent for information only and is designed to aid in maintaining current information about the total model while portions of the model are being processed through the Kit Cycle. The Update Kit may include only those pages changed since the previous updating.

Standard Kits contain portions of a model and are submitted frequently as work progresses. Standard Kits are submitted through the IDEF Kit Cycle for review and are the type referred to in the rest of this annex.

Update Kits are submitted at regular intervals. These kits contain the latest version of the model. Recipients of Update Kits are not expected to make comments on them although they may choose to do so. Update Kits are kept by the recipients for their files.

An appropriate cover sheet distinguishes the material as a kit. The cover sheet has fields for author, date, project, document number, title, status and notes. Prepare one Cover Sheet for each kit submitted, filling in the following fields on the Cover Sheet (See Figure C2).

- Working Information

  - Author or team generating the model
  - Project name and task number
  - Date of original submission to library
  - Dates of all published revisions
  - Status of model, either working, draft, recommended for acceptance or publication as final model

- Reviewer Information

  - Filing and copying information
  - List of kit reviewers
  - Schedule date for various stages of kit cycle

- Content Information

  - Table of contents for the kit
  - Status of each kit section
  - Comments or special instructions to librarian

- Identification Information

  - Model Name (in Node field)
  - Title of the model
  - C-Number

Figure C2. IDEF Kit Cover Sheet

Figure C3. IDEF KIT CONTENTS FORM

To avoid oversights, review the kit as if that were the only information available. Catch any typographical errors. Add points of clarification that come to mind as brief notes on the kit itself. Glossary definitions for terms that appear in the kit should always be appended as support material.

Gather helpful materials and append these for the reader's benefit. Never use this supplemental material to convey information which should properly be conveyed by the diagram itself. Whenever possible, use the most natural means of communication—diagrams—to show details that are important for the reader in understanding the concepts. Combine all material with a completed Cover Sheet and Kit Contents Sheet (Figure C3) and submit to the Library.

The IDEF Diagram Form (Figure C4) has minimum structure and constraints. The sheet supports only the functions important to the discipline of structured analysis. They are:

Establishment of context;

Cross-referencing between pieces of paper;

Notes about the content of each sheet.

The diagram form is a single standard size for ease of filing and copying. The form is divided into three major sections:

Working Information (top)

Message Field (center)

Identification Fields (bottom)

The diagram form should be used for everything written.

Figure C4. Standard Diagram Form

The form is designed so that the Working Information Fields at the top of the form may be cut off when a final “approved for publication” version is completed. The Identification Fields at the bottom are designed to show, one under the other, when the forms (with tops intact for tacking) are spread vertically and thumb-tacked, in top down order, on a cork board or wall. The exposed Identification Field Strips act as a thumb index arranged in Node Index (outline) order. If the two-sided page-pair publication format is followed, with the Text-and-Context page bottoms toward the binding, then their information becomes visible when the wall-mounted ancestor forms are lifted up, to see the Message Field of each diagram.

This tells who originally created the diagram, the date that it was first drawn and the project title under which it was created. The “date” field may contain additional dates, written below the original date. These dates represent revisions to the original sheet. If a sheet is re-released without any change, then no revision date is added.

This provides a check-off for reader notes written on the diagram sheet. As a comment is made on a page, the corresponding note number is crossed out. This process ensures that each reader note on a diagram is assigned a unique note number, and that the note numbers on each diagram are consecutive.

The status classifications indicate stages of approval. They are:

WORKINGThe diagram is a major change, restarts the approval sequence. New diagrams are, of course, working copy, but also it is usual for diagrams to remain Working for several revisions before advancing. A new author for a diagram usually resets the status to Working, as well.

DRAFTThe diagram is a minor change from the previous diagram, and has reached some agreed-upon level of acceptance by a set of readers. Draft diagrams are those proposed by a task leader (who may be the author). Draft diagrams may undergo further revisions, if they are included in a current kit along with other diagrams, or are brought back into consideration from some reader's file, even though they are not in the current kit. Draft status remains until the diagram is accepted by a review meeting of the technical committee or coalition.

RECOMMENDEDBoth this diagram and its supporting text have been reviewed and approved by a meeting of the technical committee or coalition, and this diagram is not expected to change.

PUBLICATIONThis page may be forwarded as is for final printing and publication.

This area is where a reader should initial and date each form.

A sketch of only the box layout of the parent diagram, with the parent box highlighted. The parent diagram's node number is written in the lower left of the Context Field (Figure C5). The box number of the parent box may be written in the highlighted box, even though it also is the last digit in the child diagram's node field entry.

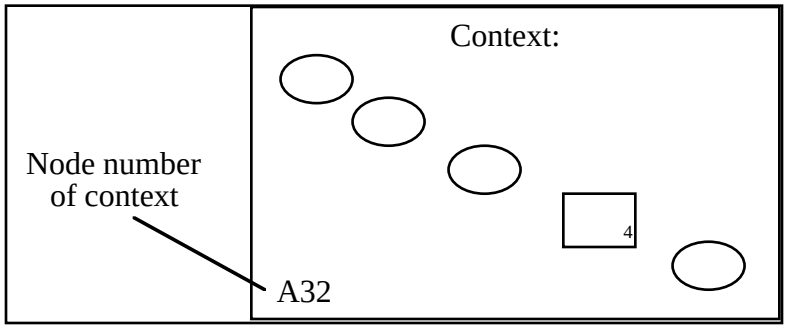


Figure C5. Illustration of context field

The following special cases arise: 1) The Context field of the required A-0 context diagram form is "TOP", written in the center of the field. 2) The Context field of the optional A-1 high-level context diagram is A-2, sketched, if there is such a higher-level context diagram, and similarly for A-n, for n = 2 or greater. 3) The Context field for the highest-level of context diagram, A-n, for largest n (= 1 or greater) is "NONE". See Figure 21 for an illustrative example.

This is a list of diagrams, other than the parent context, which use or reference this diagram-form page in some way.

The most common use is a listing of one or more node references to sub-models for which this child diagram's parent box supplies support for calls into this child's details. If necessary (the case n=1 being assumed by convention), the node_reference_expression (which begins with a "model_name/" followed by a node number) ends with "Mn", n greater than or equal to 2, making the reference into a complete ICOM-code reference to a specific Mechanism arrow of the supported sub-model's top box. For example "TLS/A34M2" written in the Used At field for child diagram "MN/A211" asserts that parent box "MN/A21" supplies mechanism support, via the second mechanism arrow of its top box, to sub-model "TLS/A34."

With such support to a remote sub-model supplied by this Used At field, then any child box on this child diagram (or more generally, even the parent box that supplies the support, or

any offspring boxes) may be called upon to supply details for some reachable offspring box of the supported box (treated as the top box of a sub-model) whose node reference appears in the Used At field. A reachable offspring box is a box that is reachable by a branching mechanism support arrow whose source is ICOM-code connected to the mechanism support.

If the top of the diagram is cut off, for publication, then the contents of the Used At field must be copied into the Message field as a -note (model note).

The Message field contains the primary message to be conveyed. The field is normally used for diagramming with an IDEF graphical language. However, the field can be used for any purpose: glossary, checklists, notes, sketches, etc. Project members should use no paper other than diagram forms, so that the reference-number-based filing system can provide a complete project record. This can be facilitated by IDEF tool support that includes E-mail and Bulletin Boarding, with automatic C-numbering and “preference”-settings for each participant.

This field contains the complete node reference for the sheet (including model_name, slash, Node Number, and “F” (for FEO), “T” (for text) or “G” (for glossary) —with the page number “1” or “2”, etc. appended at the end to indicate overflow pages, if necessary), so that the sheet is uniquely located for any and all reference purposes.

The Title field contains the name of the material presented on the Diagram Form. If the Message field contains a diagram, then the contents of the Title field must precisely match the name written in the parent box.

The large area contains the C-number. The C-number is composed of two or three letters of the author’s initials (chosen to be unique among project participants) followed by a number sequentially assigned by the author. This C-number is placed in the lower left corner of the Number field and is the primary means of reference to a sheet itself, because the sheet as a diagram form in use can only be created once. Every diagram form used by an author receives a unique C-number, which habitually should be the first mark made on the form.

If the project elects to track version history, then the C-number of the diagram form of which this sheet is an altered version shall be written parenthesized (space optional) following the author's C-number entry in the C-number field. For example “AB34(CD123)” indicates that this sheet (“AB34”) is intended to be a replacement for the already-existing

“CD123”.

When a model is published, the C-number may be replaced by a standard sequential page number (e.g., “pg. 17”).

A kit page number is written by the librarian at the right hand side of the Number field inside the small rectangle. This is composed of the document number followed by the letter identifying the sheet within the document.

Each officially-assigned participant in a project shall maintain Reader/Author files of the documents received. The librarian shall maintain the official Master and Reference files of the project, archiving each kit submitted during the course of the project.

Variations in the filing process may occur based on individual preferences, but the following files, maintained in alphabetically-sorted reference number order as the primary organizational filing structure, are the minimum:

- Standard Kit Files

  Maintained by authors, commenters and perhaps other readers. File Kit Cover Sheets chronologically, as a master log, but extract and file most C-pages in appropriate Models, in node-reference order. When in doubt, leave sheet in filed Kit, perhaps adding cross-referencing Reader Notes on already-filed diagram forms in selected other-filing places, as well. (Every sheet is the property of the Reader, and Reader-Note numbering restarts fresh for each C-numbered sheet, so the added personal filing Reader Notes cannot do any harm.)

- Updated Current Model Files:

  Maintained by authors, commenters and readers from Update Kits that are received. May be culled in favor of Project Master versions, as the project progresses.

- Working Files:

  Maintained by authors—and any Reader who initiates any ad hoc interchange between participants that has no official Author assigned. The Kit Cover Sheet for such a Reader-started topic should be suggestively named, so that an alphabetical filing of Kit contents can parallel the official Working Files of models, etc.

- Project Files:

  Maintained by the librarian to standards set by the project management.

In addition to the Kit Cycle, a Walk-Through Procedure has been developed as a guide for presenting model information to a group of “reviewers” (perhaps only weakly experienced in understanding IDEF0 models on their own). It does not substitute for the Reader/Author Cycle review process that is central to the quality assurance of IDEF0 modeling (explained in C.2), but may be streamlined for periodic project use at the technical level to provide an opportunity for all participants to share or develop common interpretations that may not surface in the one-on-one Kit-based interchanges.

1.Present the model to be analyzed by using its node index. This is the model's table of contents. Provides a quick overview of what is to come.

2.Present selected glossary terms. Encourage each reviewer to replace personal meanings of words with those that the presenting team has chosen. The meanings should not be questioned at this point. A change in meaning now would require many changes in the diagrams.

3.Present each diagram for review.

The diagram walk-through process is an orderly, step-by-step process where questions can be asked that may identify potential weaknesses in the diagram or its text. Six steps of a structured walk-through follow below.

Diagram corrections may be proposed at any step. These corrections may be noted for execution at a later date or adopted immediately.

Step 1: Scan the diagram.

This step allows the reader to obtain general impressions about the content of the diagram. Typically, the reader will have reviewed the parent diagram which depicted the current diagram as one of its boxes. The reader is now examining how the author decomposed that function.

Criteria for acceptance:

1.The decomposition is useful for its purpose and is complete within the context of its parent box. All lower level functions can clearly be categorized under each of its boxes.

2.The diagram reflects, in the reviewer’s opinion, a relevant point of view based on the purpose of the model.

3.In the opinion of the reviewer, there is enough new information provided to extend understanding of the parent box. There is not so much detail that the diagram appears complex and hard to read.

Unless a problem is rather obvious, criticism may be delayed until Step 3 below. However, first impressions should not be lost. They might be put on a blackboard or flip chart pad until resolved.

Step 2: Look at the parent.

Once the reader understands the current diagram’s decomposition, the parent diagram should be reviewed to insure compatibility.

Criteria for acceptance:

1. The decomposition covers all of the points the reviewer anticipated when reading the parent diagram.

2. Now that the decomposition of this portion of the parent diagram is revealed, the detail which the reviewer envisioned for this box should still seem correct. If not, note the missing detail.

It might be important at this step to return to the parent diagram briefly and add new -notes 

(reader notes) or embellish existing ones, based upon the added insight gained from this look at the decomposition.

Step 3: Connect the parent box and the detail diagram.

This step tests the arrow interface connections from the parent to child.

Criteria for acceptance:

1.There are no missing or extra interface arrows.

2. Boundary arrows are labeled with proper ICOM codes.

3.Child arrow labels are the same or an elaboration of its parent's matching arrow. Labels convey the correct and complete arrow contents.

4.Examination of the connecting arrows reveal no problems in the parent diagram. (An added interface may create a misunderstanding of the message conveyed by the parent.)

A clockwise tour of the four edges of the parent box, checking each arrow, will provide a methodical way to check matching of ICOM codes boundary arrows to the parent arrows.

Step 4: Examine internal arrow pattern.

The pattern of boxes and arrows constitutes the primary expression of the model being created.

Each box will be examined in node number order, and each arrow followed in ICOM order for each box. When this process is complete, the reviewers should be led through the diagram to explore the consequences of situations with which reviewers are familiar and to test the diagram's capability to simulate the relationships known to exist.

Criteria for acceptance:

1.The diagram does not look cluttered. The number of arrow crossings and bends is minimized.

2.The boxes should be balanced with regard to detail. There should be an equal amount of detail within each box. However, compromises on this criterion are acceptable for the sake of clarity.

3.The diagram should be consistent with the reviewer's experience and knowledge of the subject matter. Feedback and error conditions should be shown as the reviewer expects.

4.The level of detail of the arrows should match the level of detail of the boxes. Bundling of arrows into more general arrows should be considered.

Step 5: Read the supportive documentation.

This step examines the points that the author highlights in the text, glossary and FEOs.

Criteria for acceptance:

1.The text confirms the interpretation obtained from examining the diagram itself.

2.Normal-paths, feedback, error-handling and other features suggested by the text are found in the diagram or found in a FEO (For Exposition Only) diagram.

3.Significant diagram features uncovered during Steps 1-4 are found in the text, glossary or FEO.

4.References to the diagram are detailed enough to connect text, glossary or FEO to specific parts of a diagram.

Step 6: Set the status of the diagram.

Set the status of the diagram (as defined earlier) to:

Working
Draft
Recommended
Publication

This section contains definitions which relate to the informative annexes of this document. See Section 2 for definitions for the normative sections. A term, if defined, is defined either in Section 2 or Annex D, but not in both.

D.1 Approval Level: One of the following four words assigned to an IDEF model to indicate its relative degree of review and approval:

Working	(Lowest level)
Draft	(Next to lowest level)
Recommended	(Next to highest level)
Publication	(Highest level)

D.2 Author: The person who prepares and is responsible for any specific IDEF model or diagram.

D.3 Clustering: Boxes are split and clustered in diagramming. Clustering is the grouping of two or more boxes to form a single box. Its goal is to combine multiple functions into a single more general function.

D.4 Commenter: A reader with sufficient training in an IDEF technique to offer specific comments using the reader note numbering system and (often) referring to flaws in the application of the technique itself.

An assigned reader who shares responsibility with the author for the quality of an IDEF kit, model, diagram or other IDEF result.

D.5 Draft: See approval level.

D.6 Expert: A person familiar with a part of the real world system (or subject) being modeled. May serve as a source of information or as a reviewer of part of the model.

D.7 IDEF Role: A position in an IDEF project. See author, expert, commenter, reader and librarian.

D.8 Kit: A standardized package of diagrams containing portions of or complete-to-date models to be reviewed. See Kit Cycle.

D.9 Kit Cover Sheet: A special form used to control the routing of a kit through a kit cycle.

D.10 Kit Cycle: A formal Reader/Author Cycle procedure which uses kits for obtaining peer or expert review during model development.

D.11 Librarian: The person responsible for routing and tracking of kits and keeping orderly project files and archives.

D.12 Project Field: The field on the IDEF diagram form which records the name of the organized task for which an IDEF model is prepared.

D.13 Publication: See approval level.

D.14 Reader: A person with (limited) training in an IDEF technique, sufficient to accurately interpret syntax and basic meanings and to read and write reader notes, who sees part or all of a model.

D.15 Reader/Author Cycle: A procedure using reader notes for obtaining peer or expert review during model development.

D.16 Reader Note: A textual comment by a reader about an IDEF0 diagram. Reader notes are not published as part of the diagram, but rather are used for communication during a Reader/Author Cycle.

D.17 Recommended: See approval level.

D.18 Reviewer: A reviewer shares accountability for the utility of an IDEF kit, model, diagram or other IDEF result. Some reviewers lack Readership training, but participate in guided walk-throughs.

D.19 Split: Boxes are split and clustered in diagramming. When a parent box is detailed on a child diagram, the parent box is split into pieces, some of which may then be clustered, to form the 3 to 6 boxes on the child diagram.

D.20 Role: Same as IDEF Role.

D.21 Working: See approval level.