

**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

12. Waivers. 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 the accomplishment of the mission of an operator of a Federal computer system, 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 a written 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 published 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

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
2. Definitions	2
3. IDEF0 Models	5
3.1 Model Concepts	5
3.2 Syntax and Semantics	6
3.2.1 Syntax	6
3.2.1.1 Boxes	6
3.2.1.2 Arrows	6
3.2.1.3 Syntax Rules	7
3.2.2 Semantics	7
3.2.2.1 Box and Arrow Semantics	7
3.2.2.2 Labels and Names	9
3.2.2.3 Box and Arrow Semantic Rules	9
3.3 IDEF0 Diagrams	10
3.3.1 Types of Diagrams	10
3.3.1.1 Top-Level Context Diagram	10
3.3.1.2 Child Diagram	11
3.3.1.3 Parent Diagram	11
3.3.1.4 Text and Glossary	13
3.3.1.5 For Exposition Only Diagrams	14
3.3.2 Diagram Features	14
3.3.2.1 Arrows as Constraints	14
3.3.2.2 Activations of a Box	14
3.3.2.3 Concurrent Operation	15
3.3.2.4 Arrows as Pipelines	15
3.3.2.5 Branching Arrows	16
3.3.2.6 Inter-Box Connections	16
3.3.2.7 Boundary Arrows	18
3.3.2.8 ICOM Coding of Boundary Arrows	18
3.3.2.9 Tunneled Arrows	20
3.3.2.10 Call Arrows	22
3.3.3 Diagram Syntax Rules	22
3.3.4 Diagram Reference Expressions	23
3.3.4.1 Box Numbers	23
3.3.4.2 Node Numbers	24
3.3.4.2.1 Node Index	24
3.3.4.2.2 Node Tree	25
3.3.4.3 Node References	26

	3.3.4.4	Model Notes	26
	3.3.4.5	Reference Notation.....	27
3.4	Models.....		28
	3.4.1	IDEF0 Model Description.....	28
	3.4.2	Context Diagrams.....	28
	3.4.3	High-Level Context Diagrams	29
	3.4.4	FEOs, Text and Glossary	30
	3.4.5	Model Name.....	30
	3.4.6	Presentation Rules	30

ANNEX A - IDEF0 CONCEPTS	31
A.1 Background	31
A.2 IDEF0 Concepts	32
A.3 Discussion of Individual IDEF0 Concepts.....	33
A.3.1 Activity Modeling Graphics.....	33
A.3.2 Communication by Gradual Exposition of Detail.....	34
A.3.3 Disciplined Teamwork	36
ANNEX B - CREATING AND INTERPRETING IDEF0 DIAGRAMS	37
B.1 Reading IDEF0 Diagrams	37
B.1.1 Approaching a Model.....	38
B.1.2 Diagram Reading Steps.....	39
B.1.3 Semantics of Boxes and Arrows	41
B.1.3.1 Constraints Omit How and When	41
B.1.3.2 Multiple Inputs, Controls and Outputs.....	42
B.2 Author's Guide to Creating IDEF0 Diagrams	44
B.2.1 Basic Steps of Authoring.....	44
B.2.1.1 Selecting a Context, Viewpoint and Purpose.....	45
B.2.1.2 Creating the Context Diagram.....	45
B.2.1.3 Creating the Top-Most Diagram	45
B.2.1.4 Creating Child Diagrams.....	46
B.2.1.5 Creating Supporting Material.....	46
B.2.1.6 Selecting a Box to Detail.....	47
B.2.1.7 Author Activities	47
B.2.1.7.1 Data Gathering Phase	47
B.2.1.7.2 Structuring Phase.....	47
B.2.1.7.3 Presentation Phase.....	48
B.2.1.7.4 Interaction Phase	48
B.2.2 Drawing an IDEF0 Diagram	48
B.2.2.1 Generating Function Boxes.....	48
B.2.2.2 Creating Interface Arrows.....	49
B.2.2.3 Level of Effort.....	51
B.2.3 Re-Drawing an IDEF0 Diagram	51
B.2.3.1 Modifying Boxes.....	51
B.2.3.2 Bundling Arrows.....	52
B.2.3.3 Proposing Modifications to the Context	52
B.2.3.4 ICOM Syntax for Connecting Diagrams.....	53
B.2.4 Graphic Layout.....	53
B.2.4.1 Constraints on the Diagram.....	53

B.2.4.2	Arrow Placement.....	54
B.2.4.3	Arrow Layout.....	55
B.2.5	Writing Text.....	57
B.2.5.1	Text and Glossary.....	57
B.2.5.2	Notes and References.....	58
B.3	Data Collection for IDEF Modeling.....	60
B.3.1	Introduction.....	60
B.3.2	The Interview Process.....	60
B.3.3	The Interview Kit.....	61

1. Overview

1.1 Scope

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.

1.2 Purpose

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).

2. Definitions

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.
- 2.44 Semantics:** 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.

3. IDEF0 Models

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.

3.1 Model Concepts

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.

3.2 Syntax and Semantics

3.2.1 Syntax

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.

3.2.1.1 Boxes

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.



- Function name is a verb or a verb phrase.
- A box number is shown.

Figure 1. Box Syntax

3.2.1.2 Arrows

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.

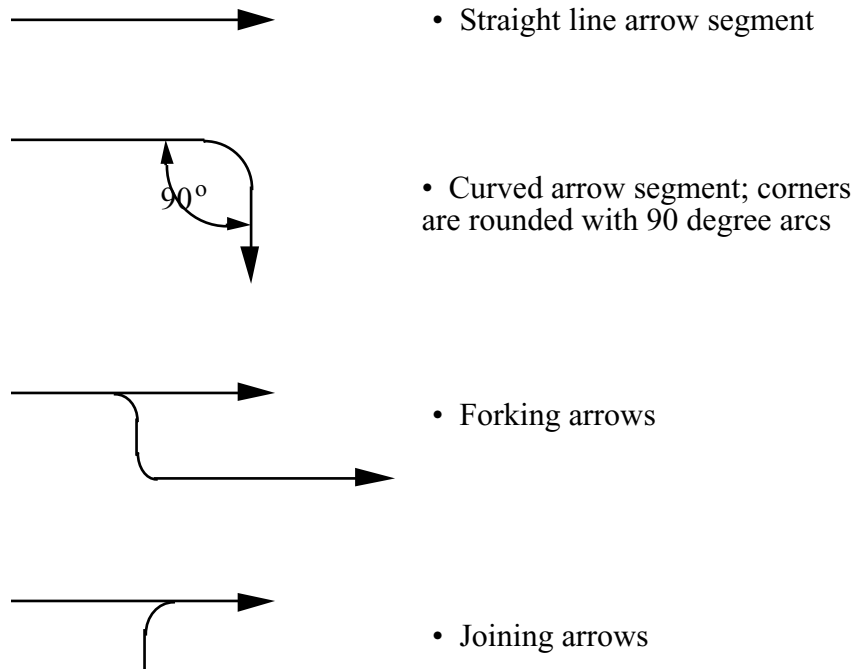


Figure 2. Arrow Syntax

3.2.1.3 Syntax Rules

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.

3.2.2 Semantics

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.

3.2.2.1 Box and Arrow Semantics

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.

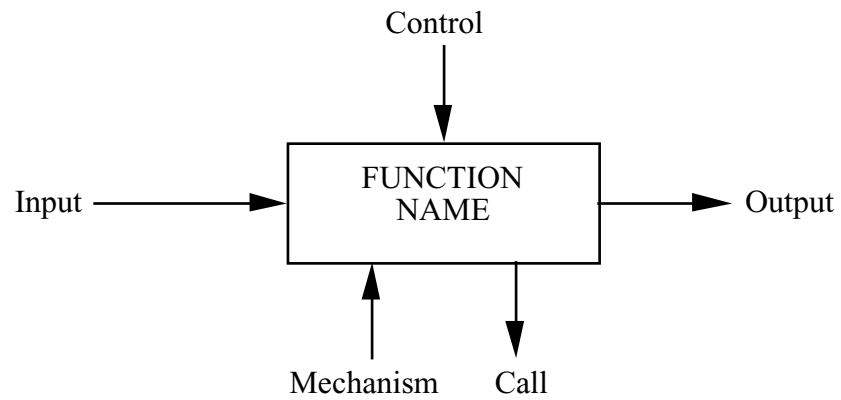


Figure 3. Arrow Positions and Roles

3.2.2.2 Labels and Names

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.

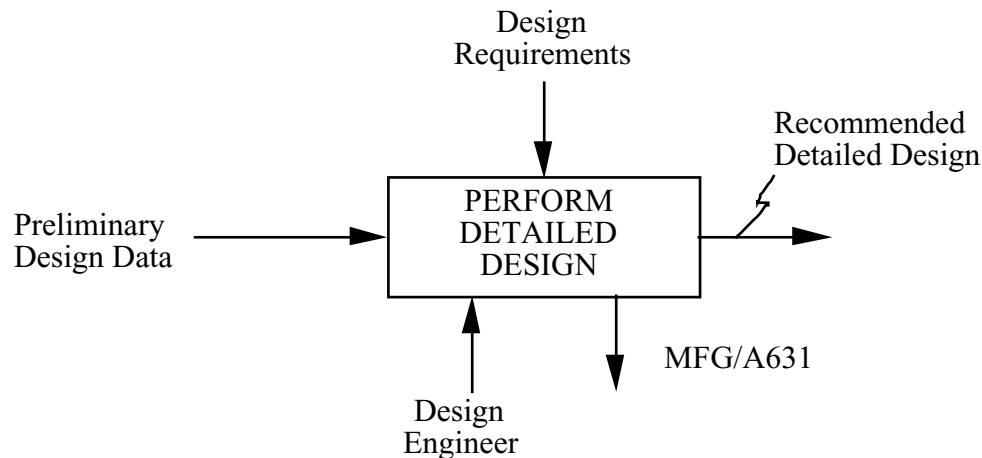



Figure 4. Label and Name Semantics

3.2.2.3 Box and Arrow Semantic Rules

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.

3.3 IDEF0 Diagrams

3.3.1 Types of Diagrams

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.

3.3.1.1 Top-Level Context Diagram

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.

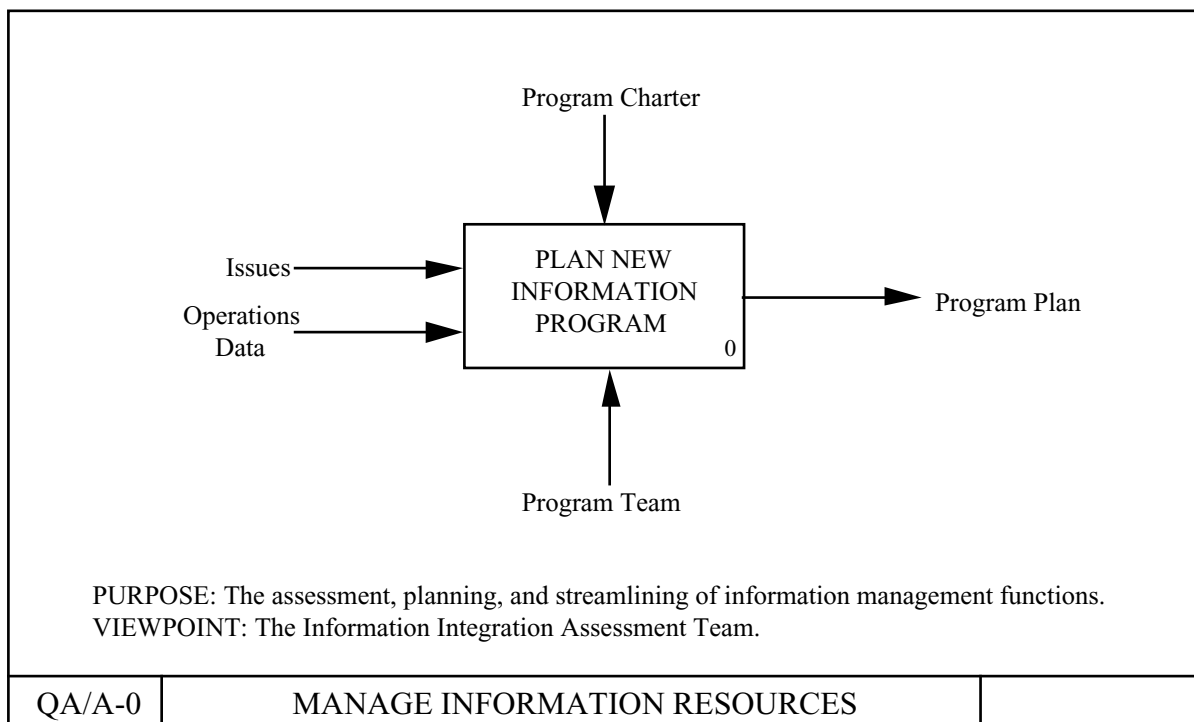


Figure 5. Example Top-level Diagram

3.3.1.2 Child 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.

3.3.1.3 Parent Diagram

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.

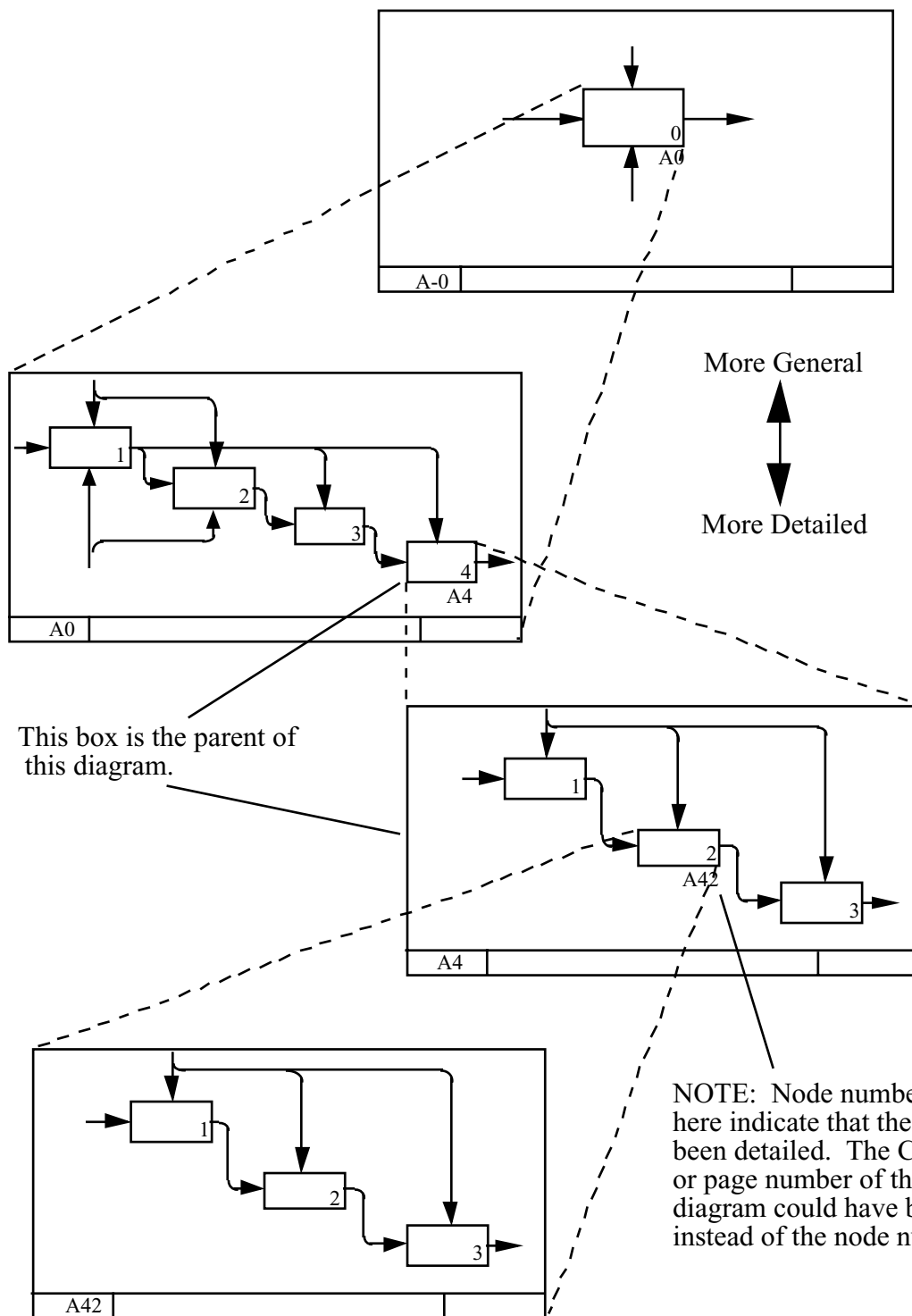


Figure 6. Decomposition Structure

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.

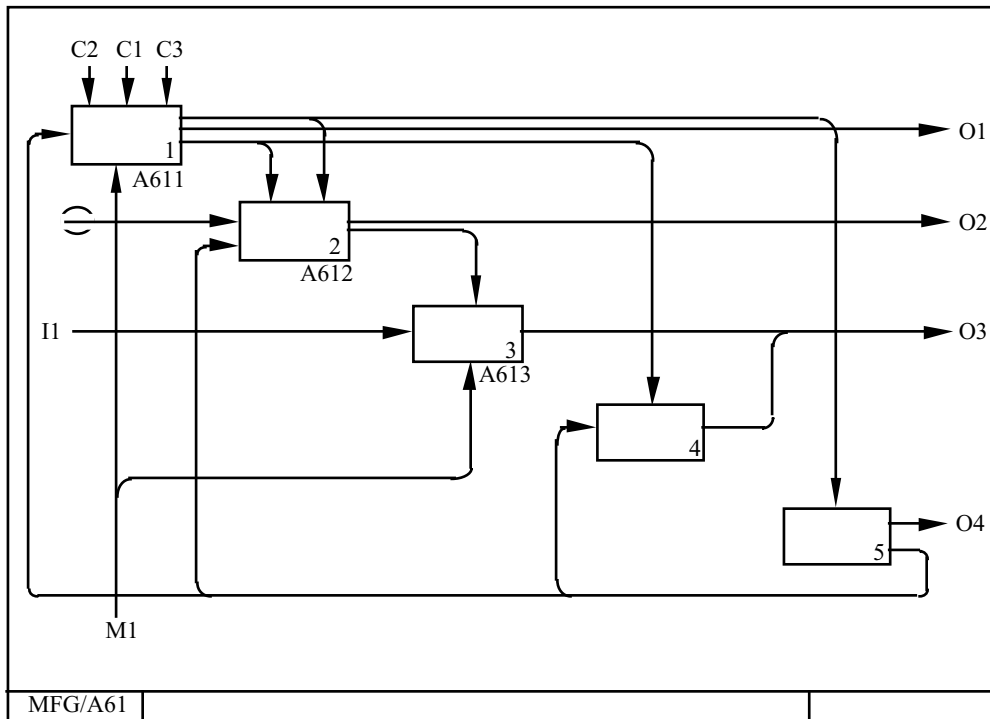


Figure 7. Detail Reference Expression Use

3.3.1.4 Text and Glossary

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.

3.3.1.5 For Exposition Only Diagrams

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.

3.3.2 Diagram Features

3.3.2.1 Arrows as Constraints

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.

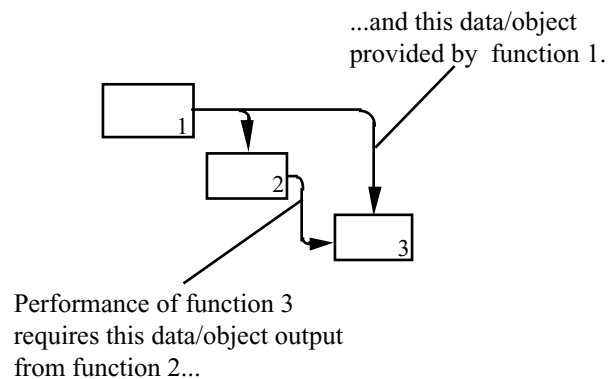


Figure 8. Meaning of Constraint

3.3.2.2 Activations of a Box

A box may perform various parts of its function under different circumstances, using different combinations of its input and controls and producing different outputs. These different performances are called the different activations of the box.

3.3.2.3 Concurrent Operation

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.

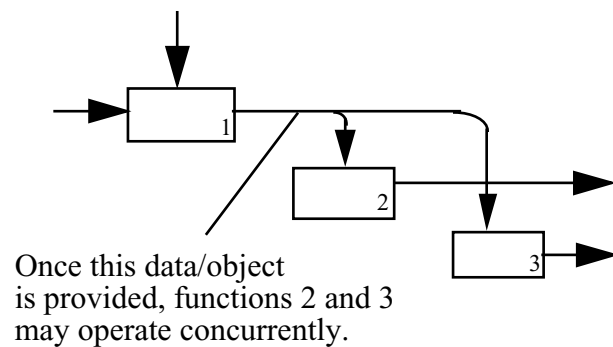
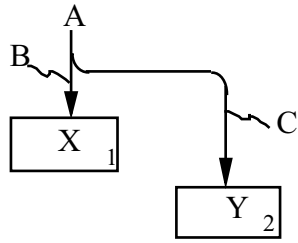


Figure 9. Concurrent Operation

3.3.2.4 Arrows as Pipelines

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.



Pipeline A splits into B and C to provide controls to X and Y.

Figure 10. Arrow Pipeline with Forking

3.3.2.5 Branching Arrows

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.

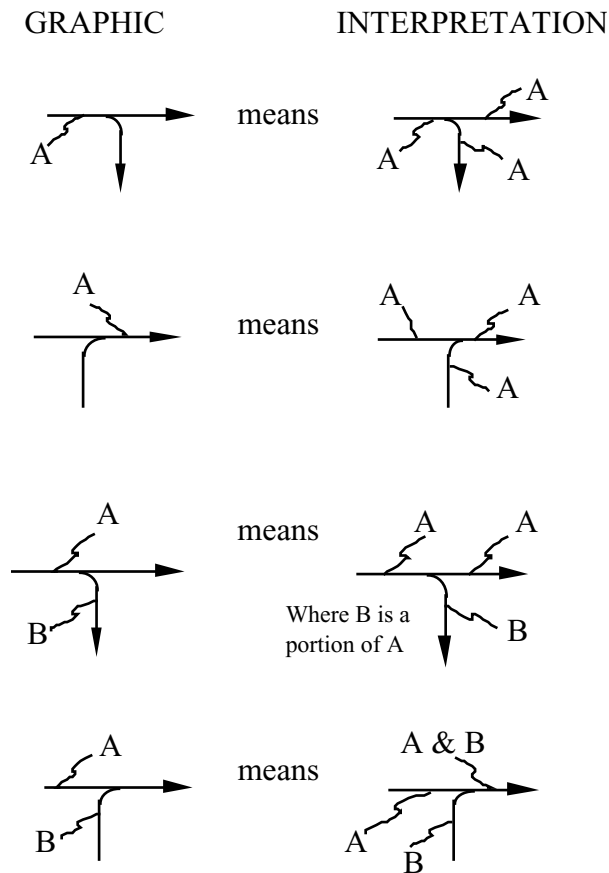


Figure 11. Arrow Fork and Join Structures

3.3.2.6 Inter-Box Connections

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.

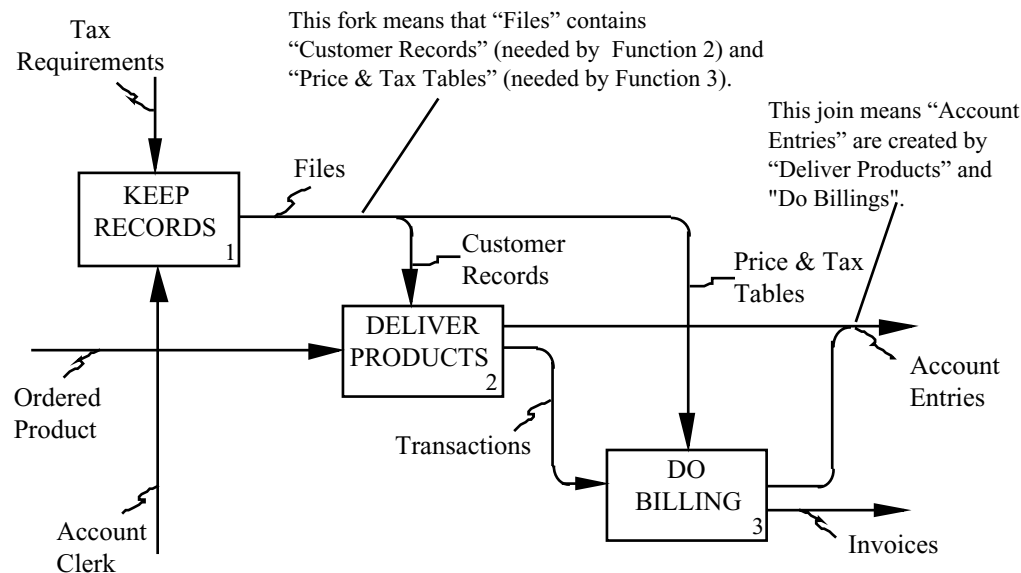


Figure 12. Connections Between Boxes

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.

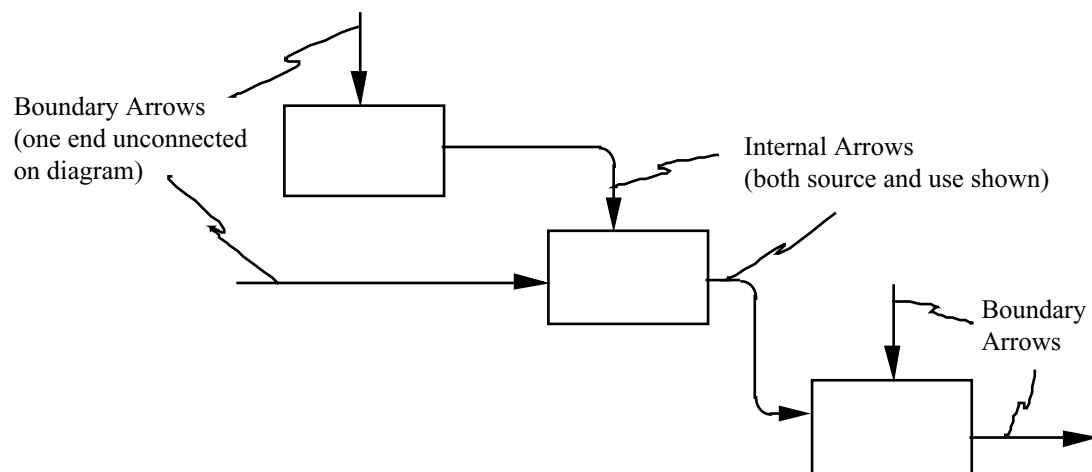


Figure 13. Boundary and Internal Arrows

3.3.2.7 Boundary 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.

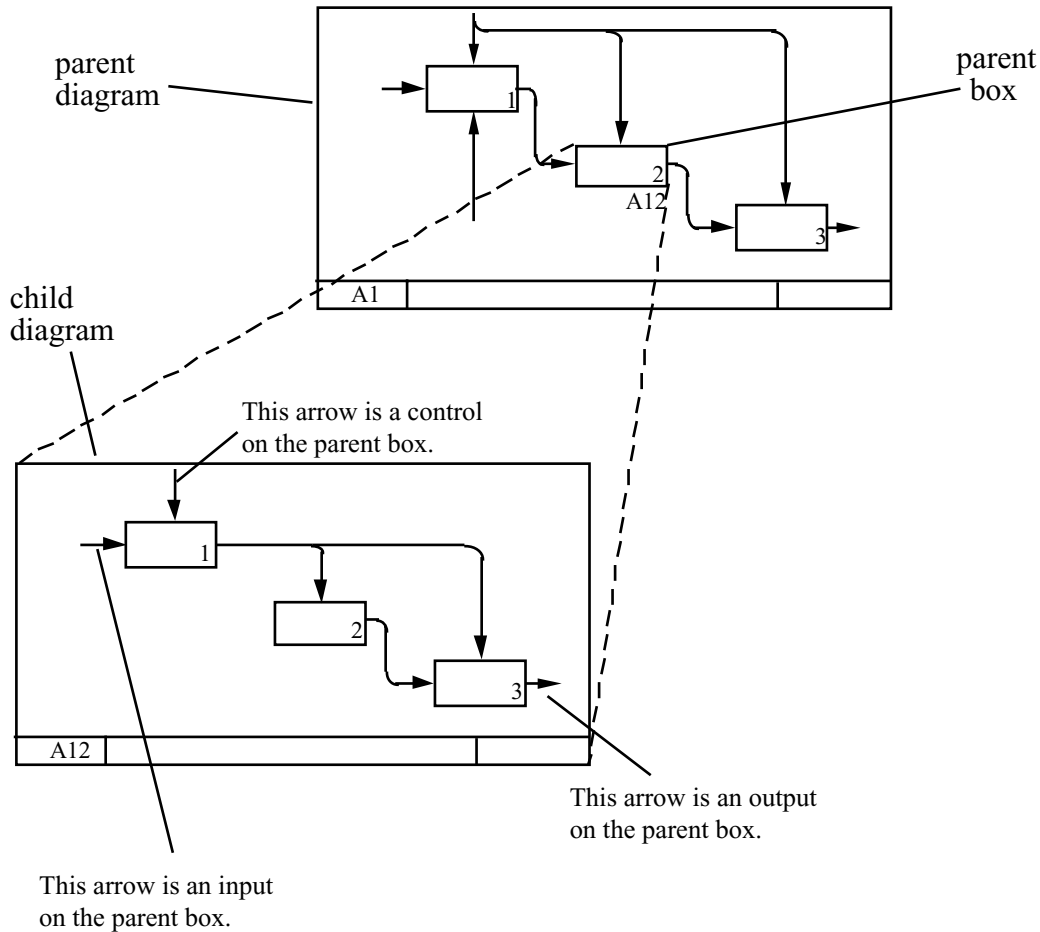


Figure 14. Boundary Arrow Correspondence

3.3.2.8 ICOM Coding of Boundary Arrows

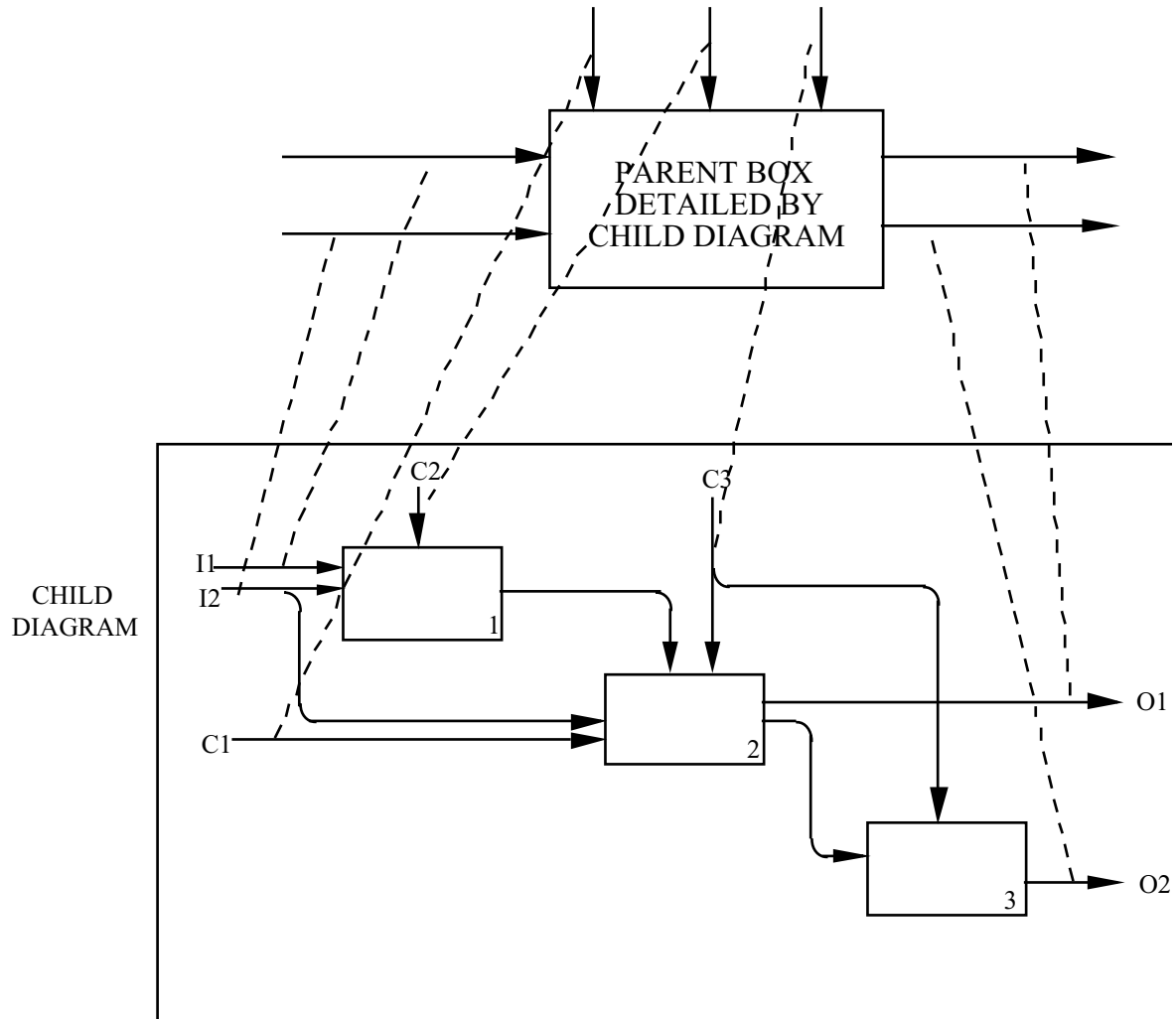
ICOM codes relate boundary arrows on a child diagram to arrows connected to its parent box. A specific notation, called ICOM codes, specifies the matching connections. The letter I, C, O or M is written near the unconnected end of each boundary arrow on the child diagram. This coding identifies the arrow 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.

Figure 15. ICOM Codes and Changing Arrow Roles



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

3.3.2.9 Tunneled Arrows

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.

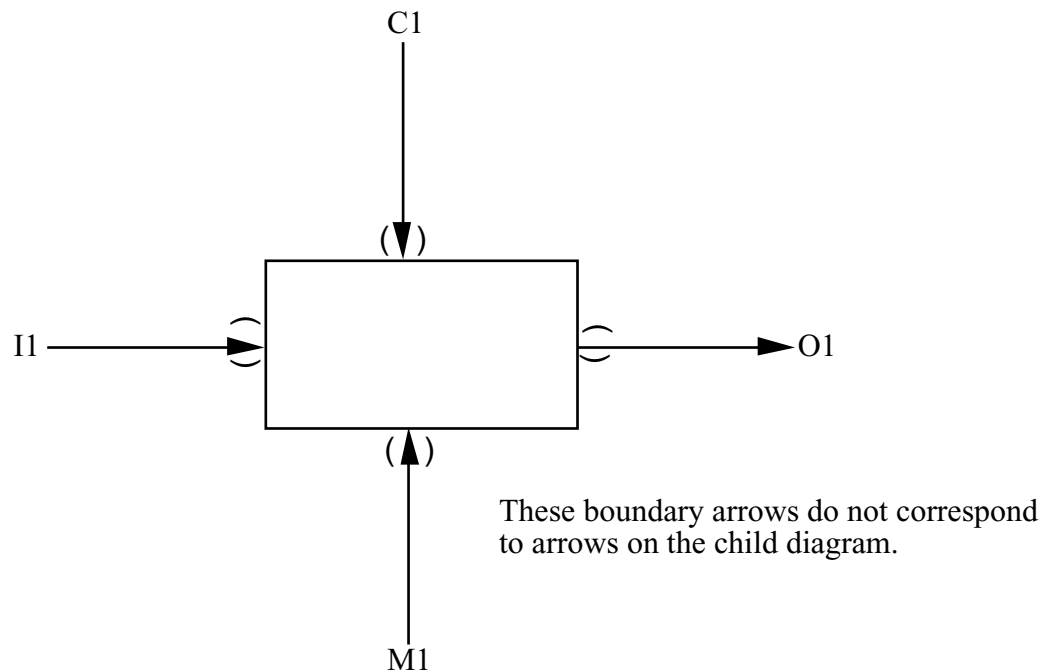


Figure 16. Arrows Tunneled at Connected End

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.

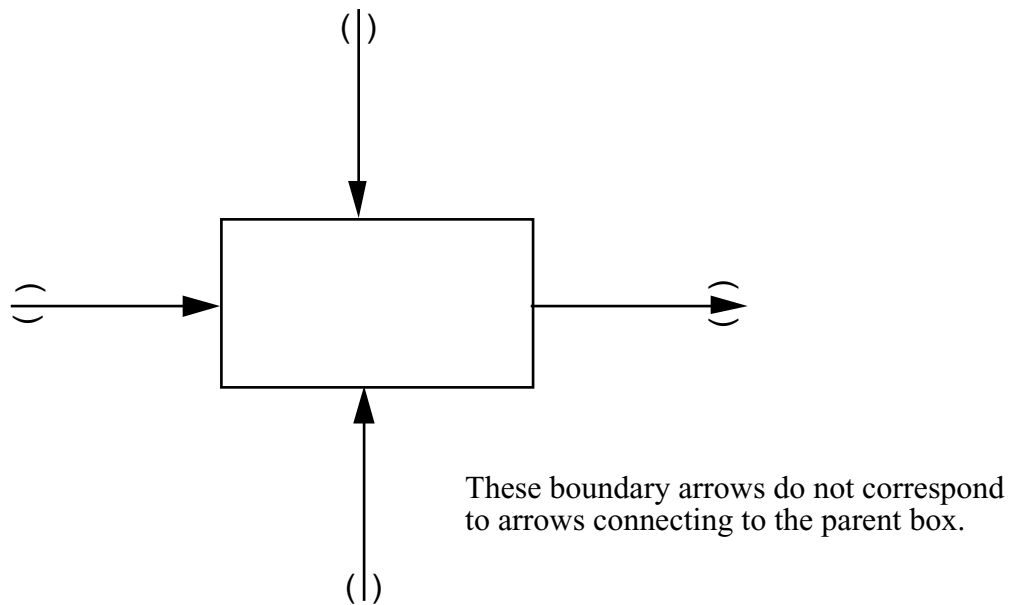


Figure 17. Arrows Tunneled at Unconnected End

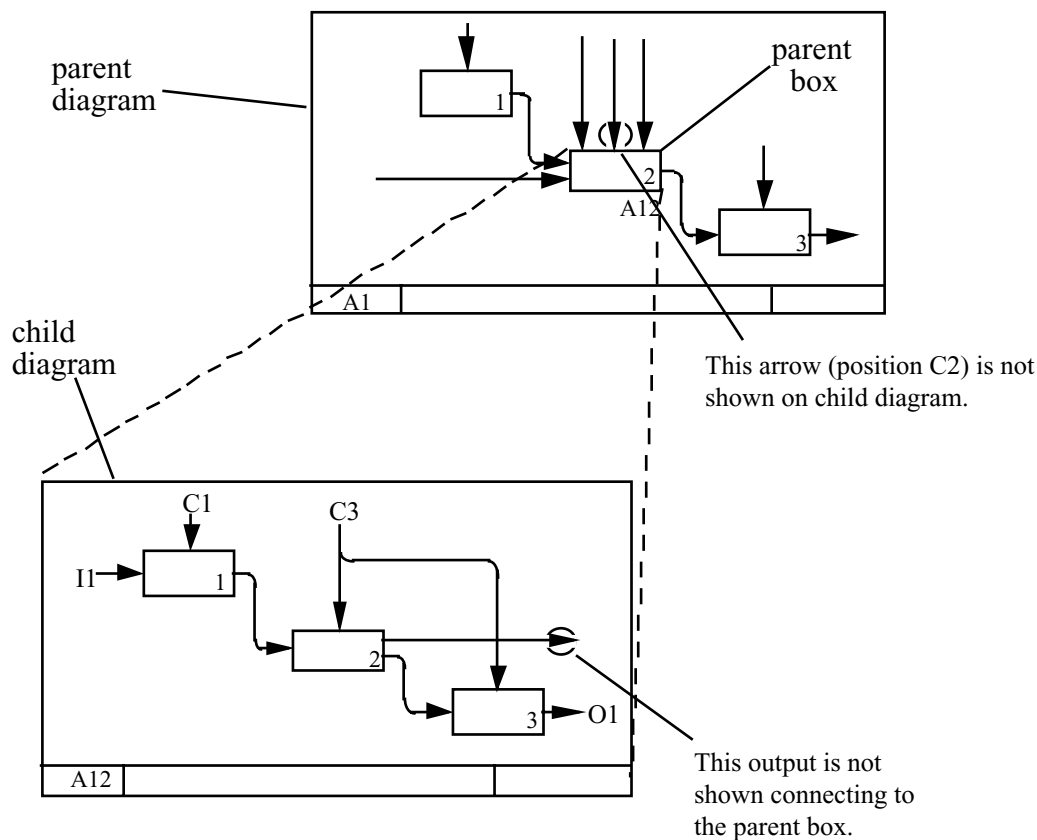


Figure 18. Example of Tunneled Arrows

3.3.2.10 Call Arrows

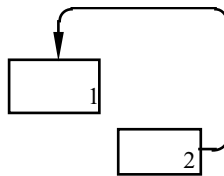
A call arrow is a special case of mechanism arrow. It signifies that the caller box does not have its own child diagram to detail it, but rather is detailed entirely by another box (and its descendants) in the same or another model. Multiple caller boxes may call the same box.

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.

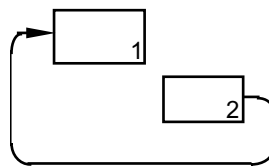
The arrows of the called box may not correspond exactly with those of the caller box, either in number or in meaning. In these cases, model notes attached to the call arrows shall specify the relationships so that the correct interpretation may be given to the shared data and objects.

3.3.3 Diagram Syntax Rules

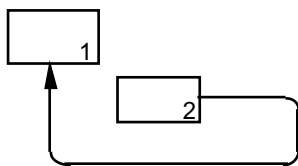
1. Context diagrams shall have node numbers A-n, where n is greater than or equal to zero.
2. The model shall contain a A-0 context diagram, which contains only one box.
3. The box number of the single box on the A-0 context diagram shall be 0.
4. A non-context diagram shall have at least three boxes 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”.



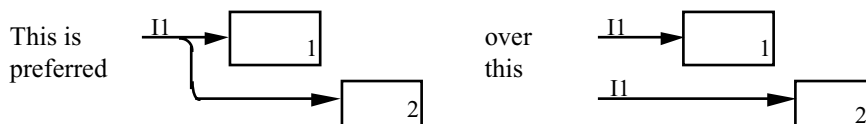
Input feedbacks shall be shown as “down and under”.



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.

3.3.4 Diagram Reference Expressions

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.

3.3.4.1 Box 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.

3.3.4.2 Node Numbers

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-1	Optional context diagram
A-0	Required 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, ..., A666	Child 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.

3.3.4.2.1 Node Index

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

3.3.4.2.2 Node Tree

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.

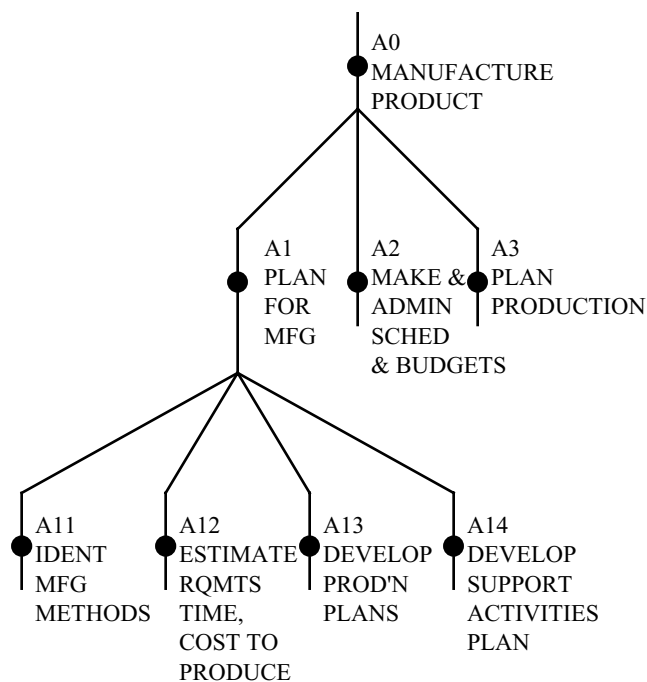


Figure 20. Typical Node Tree

3.3.4.3 Node References

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).

3.3.4.4 Model Notes

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

3.3.4.5 Reference Notation

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. 3	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.