BS ISO/IEC/IEEE 31320-1:2012

BSI Standards Publication

Information technology — Modeling Languages

Part 1: Syntax and Semantics for IDEF0

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

National foreword

This British Standard is the UK implementation of ISO/IEC/IEEE 31320-1:2012.

The UK participation in its preparation was entrusted to Technical Committee IST/15, Software and systems engineering.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2013.
Published by BSI Standards Limited 2013

ISBN 978 0 580 81152 4
ICS 35.080

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 August 2013.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

INTERNATIONAL STANDARD

ISO/IEC/IEEE 31320-1

First edition
2012-09-15

Information technology — Modeling Languages —

Part 1:
Syntax and Semantics for IDEF0

Technologies de l'information — Langages de modélisation —

Partie 1: Syntaxe et sémantique pour IDEF0

Reference number
ISO/IEC/IEEE 31320-1:2012(E)

Documents
Document Sharing Hub
© IEEE 1998

Full document details: BS ISO/IEC/IEEE 31320-1:2012

COPYRIGHT PROTECTED DOCUMENT

© IEEE 1998

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from ISO, IEC or IEEE at the respective address below.

ISO copyright office
Case postale 56
CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
E-mail inmail@iec.ch
Web www.iec.ch

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York
NY 10016-5997, USA
E-mail stds.ipr@ieee.org
Web www.ieee.org

Published in Switzerland

Documents
Document Sharing Hub
© IEEE 1998 – All rights reserved

Full document details: BS ISO/IEC/IEEE 31320-1:2012

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

The main task of ISO/IEC JTC 1 is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from ISO or the IEEE Standards Association.

ISO/IEC/IEEE 31320-1 was prepared by the Software & Systems Engineering Standards Committee of the IEEE Computer Society (as IEEE 1320.1-1998). It was adopted by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in parallel with its approval by the ISO/IEC national bodies, under the “fast-track procedure” defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE. IEEE is responsible for the maintenance of this document with participation and input from ISO/IEC national bodies.

ISO/IEC/IEEE 31320 consists of the following parts:

— ISO/IEC/IEEE 31320-1, *Information technology — Modeling Languages — Part 1: Syntax and Semantics for IDEF0*

— ISO/IEC/IEEE 31320-2, *Information technology — Modeling Languages — Part 2: Syntax and Semantics for IDEF1X$_{97}$ (IDEF$_{\text{object}}$)*

© IEEE 1998 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

(blank page)

© IEEE 1998 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

IEEE Std 1320.1-1998(R2004)

IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0

Sponsor

Software Engineering Standards Committee
of the
IEEE Computer Society

Approved 25 June 1998

IEEE-SA Standards Board

Abstract: IDEF0 function modeling is designed to represent the decisions, actions, and activities of an existing or prospective organization or system. IDEF0 graphics and accom-panying texts are presented in an organized and systematic way to gain understanding, support analysis, provide logic for potential changes, specify requirements, and support system-level design and integration activities. IDEF0 may be used to model a wide variety of systems, composed of people, machines, materials, computers, and information of all varieties and structured by the relationships among them, both automated and nonautomated. For new systems, IDEF0 may be used first to define requirements and to specify functions to be carried out by the future system. As the basis of this architecture, IDEF0 may then be used to design an implementation that meets these requirements and performs these functions. For existing systems, IDEF0 can be used to analyze the functions that the system performs and to record the means by which these are done.
Keywords: enterprise, functional modeling language, IDEF0, language, modeling language

---

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1998 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1998. Printed in the United States of America.

ISBN 0-7381-0340-3

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Copyright © 1998 IEEE. All rights reserved.
Documents Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

Introduction

(This introduction is not part of IEEE Std 1320.1-1998, IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0.)

This standard formally documents the Integration Definition 0 (IDEF0) language for function models in two parts. The body of this standard describes the syntax and semantics of the IDEF0 language that are required to draw the physical diagrams of a *specific* IDEF0 model. Annex B describes the syntax and semantics of the IDEF0 language as an *abstract* formal structure and, therefore, provides the foundation for the specifics found in the body of the standard. The diagrams discussed by the standard are real instantiations in a concrete model of the mathematical formalisms of the IDEF0 Language Abstract Formalization (the "language formalization").

Background

During the 1970s, the US 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 modeling methodologies known as the ICAM Definition (IDEF) methods, which include

a) IDEF0, used to produce a function model. A function model is a structured representation of the functions within a system or subject area.

b) IDEF1, used to produce an information model. An information model represents the structure and semantics of information within a system or subject area.

c) IDEF2, used to produce a dynamics model. A dynamics model represents the behavior of a system or subject area as it varies over time.

IDEF0 was derived from a well-established graphical modeling method known as the Structured Analysis and Design Technique (SADT). IDEF0 was developed by the originators of SADT, notably Douglas T. Ross, under the ICAM program. In 1983, the US Air Force Integrated Information Support System program enhanced the IDEF1 information modeling technique to develop IDEF1 extended (IDEF1X), a semantic data modeling method.

Continued IDEF method developments followed to address needs for additional analytic methods. These follow-on developments have been directed toward providing a mutually supportive family of methods that are applicable to a broad range of enterprise improvement and integration strategies (e.g., concurrent engineering, total quality management, business reengineering). Reflecting this general applicability, the IDEF acronym has been recast to refer to an integrated family of Integration Definition methods. 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. IDEF0 has been widely adopted as the function modeling method of choice in large number of both military and nonmilitary organizations in both North America and Europe.

In 1991, the National Institute of Standards and Technology (NIST) received support from the US Department of Defense, Office of Corporate Information Management (DoD/CIM), to develop Federal Information Processing Standards (FIPS) for modeling techniques for use by the federal government. One

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

product of this effort was FIPS PUB 183, *Integration Definition for Function Modeling (IDEF0)* [B2].[a] This FIPS document was based on the IDEF specification manuals published by the US Air Force in the early 1980s. At the same time, NIST also published FIPS PUB 184, *Integration Definition for Information Modeling (IDEF1X)* [B3], to support data modeling for the federal government.

In 1993, the Institute of Electrical and Electronics Engineers (IEEE) Computer Society initiated a project to establish IDEF standards across both industry and government within the standards framework of the American National Standards Institute (ANSI). IEEE Std 1320.1-1998 for IDEF0 function models, based on FIPS PUB 183 [B2], is a result of that effort.

This standard is explicitly oriented to the presentation of an IDEF0 model on paper pages; development of an IDEF0 standard for other presentation media is deliberately not addressed by this document. However, integrated into this standard is a mathematically founded formalization of an IDEF0 model. This formalization allows users to separate what they conceive in conceptual space, that is, the model itself, from their presentation of that model and from their presentation media. In earlier work, the IDEF0 diagram was not considered as something that should or could be distinguished from the paper page that presents that diagram. However, the current formalization allows (indeed, forces) users to separate the abstract structure of an IDEF0 model from the physical structure of the presentation of that model using sheets of paper. An important conceptual and notational difference between this work and earlier work is the clear distinction between an IDEF0 graphic diagram and the medium, e.g., a page of paper, that is used to present that diagram.

New terminology has been presented to ensure that this distinction can be easily maintained, and a more robust categorization of both the components of an IDEF0 model and of an IDEF0 diagram has been introduced. This terminology ensures that the usage presented in the body of this standard is consistent with the formalization presented in Annex B. In addition, these changes to the IDEF0 vocabulary will facilitate the development of IDEF0 presentations in digital or other media.

The IDEF0 approach

IDEF0 includes both a modeling language and a comprehensive methodology for developing models. This standard addresses only the syntax and semantics of the modeling language itself.

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.

IDEF0 function modeling is designed to represent the decisions, actions, and activities of an existing or prospective organization or system. For all its apparent simplicity, the method is surprisingly powerful and effective. Like most modeling methods, the primary component of IDEF0 is a graphical language whose constructs are intended to convey information of a certain sort. IDEF0 graphics and accompanying texts are presented in an organized and systematic way to gain understanding, support analysis, provide logic for potential changes, specify requirements, and support systems-level design and integration activities. IDEF0 may be used to model a wide variety of systems, composed of people, machines, materials, computers, and information of all varieties and structured by the relationships among them, both automated and nonautomated. For new systems, IDEF0 may be used first to define requirements and to specify functions to be carried out by the future system. As the basis of this architecture, IDEF0 may then be used to design an implementation that meets these requirements and performs these functions. For

[a]The numbers in brackets correspond to those of the bibliography in Annex A.

Copyright © 1998 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

existing systems, IDEF0 can be used to analyze the functions that the system performs and to record the 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, with accompanying explanatory text, illuminating graphical, and defining glossary pages that are cross-referenced to these diagrams. The two primary modeling components of a diagram are functions (represented by named boxes) and the physical and data objects that interrelate those functions (represented by labeled arrows).

As a function modeling language, IDEF0 has these characteristics:

a) 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.

b) It is a coherent and simple language, allowing rigorous and precise expression and promoting consistency of usage and interpretation.

c) It enhances communication among analysts, architects, developers, managers, and users through its ease of learning and its emphasis on hierarchical exposition of detail.

d) It is well-tested and proven through many years of use by the USAir Force and other government agencies and by private industry.

e) It can be generated by a variety of computer-based tools; several commercial products specifically support development and analysis of IDEF0 diagrams and models.

As a system engineering technique, IDEF0 may be used 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. When used in a systematic way, IDEF0 provides a systems engineering approach to

a) Performing systems analysis and design at all levels, including the entire enterprise, a system, or a subject area;

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

c) Allowing collaborative team consensus to be achieved by shared understanding;

d) Managing large and complex projects using qualitative measures of progress; and

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

Typographic conventions

A word that has a special meaning for IDEF0 is italicized the first time that it is used in its specific sense for IDEF0. There will be an entry in Clause 2 for each italicized word.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

Figure conventions

The figures in this document have been prepared using the IDEF standard diagram form (SDF). The SDF and its use are documented by both FIPS PUB 183 [B2] and FIPS PUB 184 [B3]. FIPS PUB 183 [B2] does not standardize the SDF; instead, the SDF is covered in an informative annex. In keeping with this precedent, this standard also refrains from addressing the SDF as an element of IDEF0 language standardization.

The IDEF SDF is designed for use and presentation on conventional 8½-by-11-inch paper sheets. Therefore, to preserve this conventional size, the figures for this document have been collected at the end of the document rather than being interspersed with the text.

Copyright © 1998 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

Contents

1. Overview 1
1.1 Scope 1
1.2 Purpose 1

2. Definitions, acronyms, and abbreviations 2
2.1 Definitions 2
2.2 Acronyms and abbreviations 8

3. References 9

4. IDEF0 models 9

5. IDEF0 syntax 9
5.1 Boxes 10
5.2 Arrows and arrow segments 10
5.3 Names and labels 10
5.4 Basic box/arrow syntax 12
5.5 Call arrows 12

6. IDEF0 semantics 13
6.1 Box/arrow semantics 13
6.2 Branching and joining arrows 15
6.3 Arrow meaning conventions 17
6.4 Ambiguous arrow segments 18
6.5 Ambiguous arrow attachments 18
6.6 Arrow role conventions 19
6.7 Activations 19
6.8 Concurrent activation 19

7. IDEF0 diagrams 20
7.1 Diagram identification 20
7.2 A-0 context diagram 20
7.3 Model name 21
7.4 Model viewpoint 21
7.5 Model purpose 22
7.6 Optional context diagrams 22
7.7 Decomposition diagrams 24
7.8 Parent/child diagram relations 24

8. IDEF0 model pages 25
8.1 Diagram pages 25
8.2 Text pages 25
8.3 Glossary pages 25
8.4 FEO pages 26
8.5 Other pages 26

9. IDEF0 diagram features 26
9.1 Boxes 26
9.2 Interbox connections 26
9.3 Boundary arrow segments 27
9.4 Tunneled arrows 28
9.5 Model notes 30

10. IDEF0 reference expressions 31
10.1 Box numbers 34
10.2 Node numbers 34

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

10.3 Diagram numbers 35
10.4 Node tree 36
10.5 Node index 37
10.6 Diagram references 37
10.7 Page references 37

11. IDEF0 diagram feature references 37

Annex A (informative) Bibliography 42

Annex B (informative) IDEF0 language: abstract formalization 43

Annex C (informative) Examples of IDEF0 usage and style 54

Annex C (informative) Examples of IDEF0 usage and style 54

Annex D (informative) IEEE list of participants 106

Copyright © 1998 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

IEEE Standard for Functional Modeling Language— Syntax and Semantics for IDEF0

1. Overview

This standard provides requirements for the construction of semantically and syntactically correct Integration Definition 0 (IDEF0) models and diagrams. A model or diagram conforms to this standard if it adheres to all mandatory provisions (marked by the verbs “shall” or “is”) of this standard.

Annex C of this document collects the figures that are referenced in the body of the standard and in Annex B.

1.1 Scope

The body of the standard describes the modeling language (syntax and semantics) that supports the IDEF0 method for developing graphical representations of a system or subject area. The clauses that follow govern the physical construction of IDEF0 models that represent functions, functional relationships, and the physical and data objects required by those relationships.

This part of the document is divided into 10 clauses. Clause 1 provides an overview of this part of the standard. Clause 2 defines key terms. Clause 3 discusses the concept of an IDEF0 model. Clause 4 defines the syntax of the IDEF0 language. Clause 5 defines the semantics of the language. Clause 6 describes the different types of IDEF0 diagrams. Clause 7 presents the different types of IDEF0 model pages. Clause 8 provides details on the various features of an IDEF0 diagram. Clause 9 defines IDEF0 reference expressions. Finally, Clause 10 defines IDEF0 diagram feature references.

Documentation of best commercial practices and guides to recommended usage are beyond the scope of this document.

1.2 Purpose

The objectives of this part of the standard are to prescribe the construction and components of an IDEF0 model and to define the correct syntax and semantics for construction of an IDEF0 diagram within an IDEF0 model.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

2. Definitions, acronyms, and abbreviations

This clause contains definitions of important IDEF0 terms used in this standard. Terms used in a definition that are defined elsewhere in this clause are italicized.

2.1 Definitions

2.1.1 A-0 context diagram: The only *context diagram* that is a required for a valid *IDEF0 model*, the *A-0* diagram contains one *box*, which represents the top-level *function* being modeled, the *inputs*, *controls*, *outputs*, and *mechanisms* attached to this *box*, the full *model name*, the *model name abbreviation*, the model's *purpose statement*, and the model's *viewpoint statement*.

2.1.2 A-0 diagram: See: *A-0 context diagram*.

2.1.3 activation: One occurrence of a *function*'s transformation of some subset of its *inputs* into some subset of its *outputs*.

2.1.4 activation constraint: A *function*'s requirement for the presence of a nonempty *object set* in a particular *arrow role* as a precondition for some *activation* of the *function*.

2.1.5 activity: See: *function*.

2.1.6 ancestral box: A *box* related to a specific *diagram* by a *hierarchically consecutive* sequence of one or more parent/child relationships.

2.1.7 ancestral diagram: A *diagram* that contains an *ancestral box*.

2.1.8 arrow: A directed line, composed of one or more connected *arrow segments* in a single *diagram* from a single source (*box* or *diagram boundary*) to a single use (*box* or *diagram boundary*). See also: *arrow segment*, *boundary arrow*, *internal arrow*.

2.1.9 arrow label: A noun or noun phrase associated with an *arrow segment* to signify the *arrow meaning* of the *arrow segment*. Specifically, an arrow label identifies the *object type set* that is represented by an *arrow segment*.

2.1.10 arrow meaning: The *object types* (e.g., a physical thing, a data element) of an *object type set*, regardless of how these *object types* may be collected, aggregated, grouped, bundled, or otherwise joined within the *object type set*.

2.1.11 arrow reference: See: *ICOM code*.

2.1.12 arrow role: The relationship between an *object type set* represented by an *arrow segment* and the *activity* represented by the *box* to which the *arrow segment* is attached. There are four arrow roles: *input*, *control*, *output*, and *mechanism*.

2.1.13 arrow segment: A directed line that originates at a *box* side, arrow *junction* (*branch* or *join*), or *diagram boundary* and terminates at the next *box* side, arrow *junction* (*branch* or *join*), or *diagram boundary* that occurs in the path of the line.

2.1.14 boundary arrow: An *arrow* with one end (source or use) not connected to any *box* in a *diagram*. Contrast: *internal arrow*.

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

2.1.15 boundary ICOM code: An *ICOM code* that maps an untunneled *boundary arrow* in a *child diagram* to an *arrow* attached to the *parent box* that is detailed by that *diagram*.

2.1.16 box: A rectangle containing a *box name*, a *box number*, and possibly a *box detail reference* andrepresenting a *function* in a *diagram*.

2.1.17 box detail reference: A square enclosure encompassing a *box number,* which indicates that the *box* is decomposed or detailed by a *child diagram*.

2.1.18 box ICOM code: An *ICOM code* that maps a tunneled *boundary arrow* to an *arrow* attached to some *ancestral box*.

2.1.19 box name: The verb or verb phrase placed inside a *box* that names the modeled *function*. A *box*takes as its box name the *function name* of the *function* represented by the *box*. See: *function name*.

2.1.20 box number: A single digit (***0***, ***1***, ***2***, **…**, ***9***) placed in the lower right corner of a *box* to uniquelyidentify that *box* in a *diagram*. The only *box* that may be numbered ***0*** is the *box* that represents the ***A0*** function in ***A-0*** and ***A-1*** *context diagrams*.

2.1.21 branch: A *junction* at which a *root arrow segment* (going from source to use) divides into two or more *arrow segments*. May denote *unbundling* of *arrow meaning*, i.e., the separation of *object types* from an *object type set*. Also refers to an *arrow segment* into which a *root arrow segment* has been divided.

2.1.22 bundle: (A) (As a verb) To combine separate *arrow meanings* into a composite *arrow meaning*, expressed by joining *arrow segments*, i.e., the inclusion of multiple *object types* into an *object type set*. (B) (As a noun) An *arrow segment* that collects multiple meanings into a single construct or abstraction, i.e., an *arrow segment* that represents an *object type set* that includes more than one *object type*.

2.1.23 call arrow: An *arrow* that enables the sharing of detail between *IDEF0 models* (linking them together) or within an *IDEF0 model*. The tail of a call arrow is attached to the bottom side of a *box*. One or more *page references* are attached to a call arrow.

2.1.24 call reference: A *page reference* attached to a *call arrow.*

2.1.25 called diagram: A *decomposition diagram* invoked by a *calling box* and identified by a *page reference* attached to a *call arrow.*

2.1.26 calling box: A *box* that is detailed by a *decomposition diagram* that is not the *box*'s *child diagram*. A *call arrow* is attached to the bottom of a calling box.

2.1.27 child box: A *box* in a *child diagram*.

2.1.28 child diagram: A *decomposition diagram* related to a specific *box* by exactly one child/parent relationship.

2.1.29 complete ICOM code: A *diagram feature reference* in which *dot notation* joins an *ICOM code* to a *diagram reference*.

2.1.30 context: The immediate environment in which a *function* (or set of *function*s in a *diagram*) operates.

2.1.31 context diagram: A *diagram* that presents the *context* of the *top-level function* of an *IDEF0 model*, whose *diagram number* is ***A-n,*** where $\boldsymbol{0 \leq n \leq 9}$. The one-box *A-0 context diagram* is a required context diagram; those with *diagram numbers* ***A-1***, ***A-2***, …, ***A-9*** are *optional context diagrams*.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

2.1.32 control: In an *IDEF0 model*, a condition or set of conditions required for a *function* to produce correct *output*.

2.1.33 control arrow: An *arrow* or *arrow segment* that expresses IDEF0 *control*, i.e., an *object type set* whose instances establish a condition or set of conditions required for a *function* to produce correct *output*. The arrowhead of a control arrow is attached to the top side of a *box*.

2.1.34 control loopback: *Loopback* of *output* from one *function* to be *control* for another *function* in the same *diagram*. Syn: *feedback*.

2.1.35 decomposition: The partitioning of a modeled *function* into its component *functions*.

2.1.36 decomposition diagram: A *diagram* that details its parent *box*.

2.1.37 descendent box: A *box* in a *descendent diagram*.

2.1.38 descendent diagram: A *decomposition diagram* related to a specific *box* by a *hierarchically consecutive* sequence of one or more child/parent relationships.

2.1.39 diagram: An instantiation of the formal diagram structure as defined by Annex B that consists only of semantically and syntactically valid IDEF0 graphical statements. Each diagram is a single unit of an *IDEF0 model* that presents the top-level *function* that is the subject of the model (the *A-0 context diagram*), presents the *context* of the subject *function* (other *context diagrams*), or presents the details of a *box* (*decomposition diagrams*).

2.1.40 diagram boundary: An edge of a *diagram* in a *diagram page*.

2.1.41 diagram feature: An element of a *diagram*. Diagram features include *boxes*, *arrow segments*, *arrow labels*, *ICOM codes*, *ICOM labels*, *model notes*, and *reader notes*.

2.1.42 diagram feature reference: An expression that unambiguously identifies a *diagram feature* within an *IDEF0 model*.

2.1.43 diagram number: That part of a *diagram reference* that corresponds to a *diagram*'s *parent function*'s *node number*. The diagram number refers to the *diagram* that details or decomposes the *function* designated by the same *node number*.

2.1.44 diagram page: A *model page* that contains a *context diagram* or a *decomposition diagram*.

2.1.45 diagram reference: An expression that unambiguously identifies a *diagram* and specifies the *diagram*'s position in a specific model hierarchy; a diagram reference is composed of a *model name abbreviation* and a *diagram number*.

2.1.46 diagram title: A verb or verb phrase that describes the overall *function* presented by a *diagram*; the diagram title of a *child diagram* is the *box name* of its *parent box*.

2.1.47 dot notation: A technique for naming that joins the name of a parent class to the name of a dependent class with the period character. For example, the *diagram feature reference* ***ABC/A31.3*** uses dot notation to join the *page reference* of the *parent diagram* ***ABC/A31*** to the *feature reference* for *box* ***3*** in that *diagram*.

2.1.48 feature reference: An expression that unambiguously identifies a *diagram feature* in a *diagram*.

2.1.49 feedback: See: *control loopback*.

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

2.1.50 FEO page: See: *For Exposition Only page*.

2.1.51 For Exposition Only page: A *model page* that contains pictorial and graphical information (in contrast to text) about a specific *diagram*. Unlike a *diagram*, the contents of a For Exposition Only page (*FEO page*) need not comply with IDEF0 rules.

2.1.52 fork: See: *branch*.

2.1.53 function: A transformation of *inputs* to *outputs*, by means of some *mechanisms*, and subject to certain *controls*, that is identified by a function name and modeled by a *box*. Syn: depending on the usage of a model, *activity*, task, process, operation.

2.1.54 function name: An active verb or verb phrase that describes what is to be accomplished by a *function*. A *box* takes as its *box name* the function name of the *function* represented by the *box*.

2.1.55 glossary: A set of definitions that includes *arrow labels* and *box names* used in an *IDEF0 model*.

2.1.56 glossary page: A *model page* that contains definitions for the *arrow labels* and *box names* in a specific *diagram*.

2.1.57 hierarchically consecutive: An unbroken unidirectional traversal of all *node*s between two specified *node*s in a tree. All *node*s between the origin and destination *node*s shall be visited during a traversal. All traversals from any *node* to its adjacent *node* shall be made in the same direction, either towards the root of the tree of towards the leaves of the tree. Typically, hierarchically consecutive is taken to imply from ancestral *node* (closer to the root) to descendent *node* (closer to the leaves).

2.1.58 ICOM code: An expression in one *diagram* that unambiguously identifies an *arrow segment* in another *diagram*. An ICOM code is used to associate a *boundary arrow* of a *child diagram* with an *arrow* attached to an *ancestral box*. Syn: *arrow reference*.

2.1.59 ICOM label: An *arrow label* attached without a *squiggle* directly to the arrowhead of an *output boundary arrow* or to the arrowtail of an *input*, *control*, or *mechanism boundary arrow*. An ICOM label associates a *boundary arrow* of a *child diagram* with an *arrow label* of an *arrow* attached to an *ancestral box*.

2.1.60 IDEF0 model: Abstractly, a hierarchical set of IDEF0 diagrams that depict, for a specific purpose and from a specific viewpoint, the functions of a system or subject area, along with supporting glossary, text, and For Exposition Only (FEO) information.

Concretely, a set of *model pages* that include at least an *A-0 context diagram* and an ***A0*** *decomposition diagram*, a *glossary* or specific *glossary pages*, one or more *text pages* to accompany each *diagram*, and *FEO pages* and *model pages* of other types as needed.

2.1.61 identifier: Within an *IDEF0 model*, a *model name*, a *box name*, or an *arrow label*.

2.1.62 input: In an *IDEF0 model*, that which is transformed by a *function* into *output*.

2.1.63 input arrow: An *arrow* or *arrow segment* that expresses IDEF0 *input*, i.e., an *object type set* whose instances are transformed by a *function* into *output*. The arrowhead of an input arrow is attached to the left side of a *box*.

2.1.64 input loopback: *Loopback* of *output* from one *function* to be *input* for another *function* in the same *diagram*.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

2.1.65 internal arrow: An *arrow* connected at both ends (source and use) to a *box* in a *diagram*. Contrast: *boundary arrow*.

2.1.66 join: A *junction* at which an *arrow segment* (going from source to use) merges with one or more other *arrow segments* to form a *root arrow segment*. May denote *bundling* of *arrow meanings*, i.e., the inclusion of multiple *object types* within an *object type set*.

2.1.67 junction: A point at which either a *root arrow segment* divides into *branching arrow segments* or *arrow segments join* into a *root arrow segment*.

2.1.68 leaf diagram: A *diagram* that has no *descendent diagrams*, i.e., a *diagram* that does not contain any *function* that has been decomposed.

2.1.69 leaf node: A *function* that is not decomposed. A *box* that represents a leaf node does not have a *box detail reference*.

2.1.70 loopback: An *internal arrow* that is the *output* of a *box* whose *box number* is greater than the *box number* of the *box* that uses that *arrow* as *input*, *control*, or *mechanism*. These uses are referred to as *input loopback*, *control loopback*, and *mechanism loopback*, respectively.

2.1.71 mechanism: In an *IDEF0 model*, the means used by a *function* to transform *input* into *output*.

2.1.72 mechanism arrow: An *arrow* or *arrow segment* that expresses IDEF0 *mechanism*, i.e., an *object type set* whose instances are used by a *function* to transform *input* into *output*. The arrowhead of an mechanism arrow is attached to the bottom side of a *box*.

2.1.73 mechanism loopback: *Loopback* of *output* from one *function* to be *mechanism* for another *function* in the same *diagram*.

2.1.74 model hierarchy: The *diagrams* that correspond to the *node*s of the hierarchical graph structure of an *IDEF0 model*.

2.1.75 model name: A unique, descriptive name that distinguishes one *IDEF0 model* from other *IDEF0 model*s with which it may be associated. An *IDEF0 model*'s model name and *model name abbreviation* are placed in the *A-0 context diagram* along with the model's *purpose statement* and *viewpoint statement*.

2.1.76 model name abbreviation: A unique short form of a *model name* that is used to construct *diagram references*.

2.1.77 model note: A textual and/or graphical component of a *diagram* that records a fact not otherwise depicted by a *diagram*'s *boxes* and *arrows*.

2.1.78 model note number: An integer number, placed inside a small square, that unambiguously identifies a *model note* in a specific *diagram*.

2.1.79 model page: A logical component of an *IDEF0 model* that can be presented on a single sheet of paper. Model pages include *diagram*, *text*, *FEO*, and *glossary page*s.

2.1.80 node: A modeled *function* located within the hierarchical graph structure of an *IDEF0 model* by its designated *node number*; as a *function*, a node is represented in a *diagram* by a named *box*.

2.1.81 node index: A text listing, often indented, of the *node*s in an *IDEF0 model*, shown in outline order. Same meaning and *node* content as a *node tree*.

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

2.1.82 node letter: The letter that is the first character of a *node number*.

2.1.83 node number: An expression that unambiguously identifies a *function*'s position in a *model hierarchy*. A node number is constructed by concatenating a *node letter*, the *diagram number* of the *diagram* that contains the *box* that represents the *function*, and the *box number* of that *box*.

2.1.84 node tree: A graphical listing of the *node*s of an *IDEF0 model*, showing parent-child relationships as a graphical tree. Same meaning and *node* content as a *node index*.

2.1.85 object: A member of an *object set* and an instance of an *object type*. An *object* represents something in the observable world that may be distinguished from other instances of its *object type* and may be uniquely identified.

2.1.86 object set: A subset of instantiations from the set of all possible instantiations of all *object types* within an *object type set*. An object set is a subset of the union of the members of an *object type set*; the set of *object sets* includes the empty set and the set of the union of the members of the *object type set* itself. An object set is modeled by an *arrow segment*.

2.1.87 object type: The set of all possible instantiations of a singular concept, either physical or data, within an *IDEF0 model*. An IDEF0 object type is generally analogous to an IDEF1X entity or an IDEF1 entity class.

2.1.88 object type set: A named set of one or more *object types*. An object type set may include *object types* that are themselves grouped as object type sets. An object type set is designated by an *arrow label*.

2.1.89 output: In an *IDEF0 model*, that which is produced by a *function*.

2.1.90 output arrow: An *arrow* or *arrow segment* that expresses IDEF0 *output*, i.e., an *object type set* whose instances are created by a *function* by transforming the *function*'s *input*. The arrowtail of an output arrow is attached to the right side of a *box*.

2.1.91 page reference: An expression that unambiguously identifies a *model page*. The page reference incorporates a *diagram reference* to the associated *diagram*, the type of page, and any sequencing data needed to distinguish different pages of the same type that are associated with the same *diagram*.

2.1.92 page type letter: The uppercase letter in a *page reference* that denotes a specific type of *model page*.

2.1.93 parent box: An *ancestral box* related to its *child diagram* by exactly one parent/child relationship, that is, a *box* detailed by a *child diagram*. The existence of this *child diagram* is indicated by a *box detail reference*.

2.1.94 parent diagram: A *diagram* that contains a *parent box*.

2.1.95 parent function: A *function* modeled by a *parent box*.

2.1.96 purpose statement: A brief statement of the reason for an *IDEF0 model*'s existence that is presented in the *A-0 context diagram* of the model.

2.1.97 reader note: A comment made by a reader about an *diagram* and placed on the *diagram page*. A reader note is not part of the *diagram* itself, but rather is used for communication about a *diagram* during model development.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

2.1.98 reference expression: An expression that uniquely identifies a *box*, a *node* or *function*, a *diagram*, or a *model page* within an *IDEF0 model*.

2.1.99 root arrow segment: The *arrow segment* of a *junction* from which other *arrow segment*s branch or to which other *arrow segment*s join. Syn: *root* or *root segment*.

2.1.100 root segment: See: *root arrow segment*.

2.1.101 squiggle: A short "s"-shaped line attached at one end to an *arrow label* and at the other end to an *arrow segment*. A squiggles binds an *object type set* (*arrow label*) to an *object set* (*arrow segment*).

2.1.102 text page: A *model page* that contains textual material related to a specific *diagram*.

2.1.103 top box: The *box* in the *A-0 context diagram* that models the *top-level function* of an *IDEF0 model*.

2.1.104 top-level function: The *function* modeled by the single *box* in the *A-0 context diagram* of an *IDEF0 model*.

2.1.105 tunneled arrow: An *arrow* left undrawn between its attachment to an *ancestral box* and its appearance as a *boundary arrow* on some *hierarchically consecutive descendent diagram*.

2.1.106 tunneling: The act of applying *tunnel notation* to an *arrow segment*.

2.1.107 tunnel notation: A pair of short shallow arcs, resembling a pair of left and right parentheses characters, that bracket the arrowhead or the arrowtail of an *arrow segment*.

2.1.108 unbundle: The separation of *arrow meanings*, expressed by branching *arrow segments*, i.e., the separation of *object types* from an *object type set*.

2.1.109 viewpoint statement: A brief statement of the perspective of an *IDEF0 model* that is presented in the *A-0 context diagram* of the model.

2.2 Acronyms and abbreviations

These short forms of words, names, and expressions are used in this document with the following meanings:

DRE	detail reference expression
ER	Entity relationship
FEO	For Exposition Only
ICOM	Input, control, output, and mechanism
IDEF	Integration Definition or ICAM Definition Language
IDEF0	IDEF for function modeling
IDEF1X	IDEF for data modeling
iff	if and only if
KIF	Knowledge interchange format
SDF	IDEF standard diagram form

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

3. References

No normative references are required for use with this standard. A bibliography of documents referenced in the body of the standard is provided in Annex A.

4. IDEF0 models

This clause and the clauses that follow identify the basic components of IDEF0 syntax (the drawn, visual elements of the language and how they may be used together) and IDEF0 semantics (what it means when the visual elements are used together in specific, allowable ways), specify the rules that govern the use of these modeling components, and describe the types of diagrams used in an IDEF0 model. Although the components of syntax and semantics are very highly interrelated, they are discussed separately without regard for the actual sequence of construction.

An *IDEF0 model* is a representation of a set of components of a system or subject area. Systems modeled by IDEF0 techniques are composed of interdependent parts that work together to perform an intended function. The system components represented in an IDEF0 model can be any combination of things, including people, information, software, functions, equipment, products, and raw materials. The IDEF0 model reflects how system functions interrelate and operate just as the blueprint of a product reflects how the different pieces of the product fit together.

An IDEF0 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. An IDEF0 model is composed of a hierarchical series of diagrams with associated explanatory material that gradually introduce increasing levels of detail to describe functions and their interfaces within the context of a system.

An IDEF0 model contains four primary types of model pages: diagram, text, For Exposition Only (FEO), and glossary. Diagram pages contain diagrams that define functions and functional relationships in accordance with IDEF0 box and arrow syntax and semantics. Text, FEO, and glossary pages provide additional information in support of these diagrams. This structure provides the reader with a well-bounded topic with a manageable amount of detail to learn from a single diagram and with manageable amounts of supplementary information to learn from each of the diagram's supporting pages.

Concretely, an IDEF0 model shall contain a set of model pages that includes diagram pages that present at least the model's ***A-0*** context diagram and its ***A0*** decomposition diagram, one or more glossary pages for each diagram page, and one or more text pages for each diagram page. A glossary for the entire model may be used in lieu of individual glossary pages. FEO pages and model pages of other types defined for a particular modeling project may be included as needed.

5. IDEF0 syntax

This clause defines the visual aspects of boxes and their semantic interpretation as functions; the visual aspects of arrows and their semantic interpretation as objects, including data objects; and the visual aspects of attaching arrows to boxes and the semantic interpretation of these attachments as functional interrelationships.

The basic components of a language (i.e., the lexicon) and the rules that define how those basic components can be combined to form more complex syntactic structures (i.e., the grammar) together constitute the syntax of the language. The basic components of IDEF0 syntax are boxes and arrow segments.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

5.1 Boxes

Boxes are the verbs of IDEF0 syntax: a *box* models a function in an IDEF0 model. A box shall be a rectangle with square corners. A box shall have a *box name* that is fully contained within its boundaries. The box name shall be an active verb or verb phrase that describes what must be accomplished by the function represented by the box. A box shall also contain a *box number* in its lower right corner. If a box has been detailed by a decomposition diagram, the box number shall be framed by an enclosed corner; this framing of a box number to indicate decomposition is known as a *box detail reference*[1]. A typical decomposed box is shown in Figure 1.

5.2 Arrows and arrow segments

Arrows are the nouns of IDEF0 syntax: an *arrow* models physical and data objects in an IDEF0 model. (The meaning of an arrow segment is further clarified in Clause 6.) An arrow is composed of one or more directed line segments, known as *arrow segments*, with a terminal arrowhead at one end of the arrow. As shown in Figures 2 through 5, an arrow segment shall be drawn as one or more straight horizontal and vertical line segments; an arrow may not be drawn diagonally. Horizontal and vertical line segments of an arrow segment shall be connected by a curved line segment with a 90° arc. Arrows may have branching and joining configurations.

An arrow is a linear configuration, in the sense that every two distinct arrow segments in an arrow are linked by some finite number of arrow segments. Thus, in Figure 5, the arrow ***I2:2I2*** is composed of the arrow segments labeled *Untrained Staff* and *Untrained Fitters* and the arrow segments *Trained Hefters* and *Trained Staff* constitute the distinct arrow ***3O1:O2***.

An *arrow label* consists of a single noun or noun phrase that designates the *object type set* of an arrow segment, that is, the physical and data objects represented by the arrow segment. Every arrow segment shall be labeled with an arrow label unless a single arrow label clearly applies to the arrow as a whole. Call arrows shall be labeled with one or more call references. A *squiggle* shall be used to link an arrow segment to its associated arrow label, unless a relationship between a label and a specific arrow segment is visually obvious. An arrow label that identifies a boundary arrow segment is attached to the arrowhead of an output boundary arrow or to the arrowtail of an input, control, or mechanism boundary arrow; such an arrow label is known as an *ICOM label*. Because the arrowhead or arrowtail of a boundary arrow is directly attached to the arrow label, the relationship between this label and the boundary arrow segment is always visually obvious and thus shall be drawn without a squiggle.

5.3 Names and labels

An IDEF0 model is identified by a model name, an IDEF0 box is identified by a box name, and an IDEF0 arrow segment is identified by an arrow label. These identifiers shall contain only alphanumeric characters, i.e., the letters (“A” through “Z” and “a” through “z”), the digits (“0” through “9”), the space character (“ ”), and the hyphen character (“-”, as in “well-defined”). Punctuation marks and other special characters shall not appear in identifiers. The only exceptions to this prohibition are characters required to represent a proper name such as the citation of a specific reference (e.g., *Roberts’ Rules of Order*, *DoD 8320.1-M-1*).

An identifier shall be written in title case, i.e., the first letter of each word shall be capitalized. The individual words of an identifier shall be separated by a single space; other characters (e.g., underscore (“_”), hyphen) shall not connect the words of an identifier to form a single token.

---

[1]A box detail reference may be associated with a *detail reference expression* (DRE) during the process of developing an IDEF0 model. A correct IDEF0 model does not require DREs.

Copyright © 1998 IEEE. All rights reserved.
Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

Abbreviations shall not appear in an identifier. Prepositions (e.g., *for*, *to*, *from*), conjunctives (e.g., *and*, *or*), and articles (e.g., *the*, *an*) shall not appear in an identifier, unless such a word is an essential element of a term of art customarily used by a model's intended audience and the term of art is separately defined in the model's glossary.

An identifier

— Shall express a singular concept (hence the prohibition of conjunctives)

— Shall express the nature of the model component it identifies

— May not express a relationship between a model component and any other component(s) of a model (hence the prohibition of prepositions)

— May not refer to a specific instance of an object type or to a particular activation of a function (hence the prohibition of articles)

No identifiers for different arrows or boxes shall be identical.

(In the special case, limited to control and mechanism arrows that do not change roles, and where the object type set designated by an arrow label contain only one object type, and where the set of instantiations of the object type contains only one possible instance, and thus where the set of object sets consists only of the empty set and a set containing just this one instance, only one particular object can be instantiated from the object set, the object type, and the object type set. In this case, the name of the object type set is equivalent to the name of the instantiated object. Nonetheless, the arrow label still identifies the object type set represented by the arrow segment, not the single instance of the object set modeled by the arrow segment.)

a) *Names for boxes.* A box name shall identify the transformation represented by a box, i.e., the modeled function. A box name shall be an active verb or verb phrase such as

— Order Parts

— Monitor Performance

— Develop Detail Design

— Plan Resources

— Design System

— Fabricate Component

A box name shall not contain any of these words: function, activity, or process.

b) *Labels for arrows.* An arrow label or ICOM label shall identify the object type set of an arrow segment. An arrow segment shall be labeled with a noun or noun phrase such as

— Specifications

— Performance Requirements

— Software Engineer

— Summary Report

— Architectural Design

— Board Assembly

An arrow label shall not consist solely of any of these words: input, control, output, mechanism, call, object, or data.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

Figure 6 illustrates the placement of arrow labels and box names.

5.4 Basic box/arrow syntax

More complex IDEF0 syntactic structures are formed by connecting boxes with arrows in various ways. Relative to a given box in a more complex diagram structure, there are five ways that an arrow may be attached to a box in an IDEF0 diagram. An arrow may be attached by its arrowhead to the top, to the bottom, or to the left side of a box. An arrow may be attached by its arrowtail to the bottom or to the right side of a box.

An arrow attached by its arrowhead to the left side of a box is called an *input arrow*. An arrow attached by its arrowhead to the top side of a box is called a *control arrow*. An arrow attached by its arrowhead to the bottom side of a box is called a *mechanism arrow*. An arrow attached by its arrowtail to the right side of a box is called an *output arrow*. An arrow attached by its arrowtail to the bottom side of a box is called a *call arrow*. These are the only valid ways to attach an IDEF0 arrow to an IDEF0 box. This basic box/arrow syntax is illustrated in Figures 7 through 9.

At least one control arrow and at least one output arrow shall be attached to a box. One or more input arrows and one or more mechanism arrows may be attached to a box. Only one call arrow may be attached to a box.

5.5 Call arrows

An important syntactic convention is the *call arrow*. Call arrows do not play a specific semantic role in an IDEF0 diagram; hence, they do not appear in Annex B. Rather, a call arrow allows a *calling box* to be detailed (i.e., an activity to be decomposed) by a diagram that is not an immediate child diagram of the calling box. A *called diagram* is a decomposition diagram that details a calling box. A calling box is said to *invoke* a called diagram. In other words, the details of an activity represented by a calling box are specified in a called diagram that is the immediate child of some other activity within the same model or possibly in another model entirely. A *call reference* is a page reference provided by a call arrow that identifies such a called diagram for a calling box.

a) *Model integration.* The call arrow convention allows the logical integration of models containing hierarchically related functions without requiring their integration into a single physical model. A call reference to a diagram in another model effectively links the model that contains the calling box to the model that contains the called diagram. For example, a call reference might link a box in a high-level context diagram of one model to the A0 diagram of another model to treat the subject matter of the linked activity.

b) *Shared detail.* Because more than one calling box may invoke the same called diagram via a call reference, the call arrow convention allows

   — Distinct functions with similar topologies to be modeled by a single generalized decomposition or

   — The same function to be identified in different places within the same model but to be only detailed in one place in the model's decomposition hierarchy

   This approach allows the sharing of decomposition detail by activities represented in different models and/or by activities within the same model. For example, the activities ***A323*** *Invite General Specifications Committee* and ***A5332*** *Invite Detail Specifications Committee* might both be detailed by diagram ***A5332***. A call arrow with the call reference ***A5332*** might be attached to box ***3*** in diagram ***A32***.

c) *Alternate activations.* Because a single calling box may invoke more than one called diagram via call references, the call arrow convention allows a set of detail diagrams to represent each different activation of a calling box. Each possible activation may be detailed by a called diagram that has

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

been prepared specifically to detail the activity under that particular activation, which may have been created in a different model or which details some activity elsewhere in the same model.

d) *Alternate representations.* Similarly, because a single calling box may invoke more than one called diagram via call references, the call arrow convention allows one function to be modeled by a set of alternative or variant detail diagrams. Each of these alternative decompositions may be represented by a diagram in a different model or by a detail diagram elsewhere in the same model.

A call arrow is a straight, single-segment arrow whose tail is attached to the bottom side of a calling box. Only one call arrow may be attached to a calling box. A call arrow is attached to the right of any mechanism arrows; mechanism arrows are always attached to the bottom side of a calling box to the left of any call arrow.

A call reference is a page reference that is attached to a call arrow to identify a called diagram that may be invoked by the calling box. Unlike arrow labels, a call reference may not be attached to the call arrow by a squiggle. A call reference shall be placed immediately above the back of the call arrow's arrowhead and immediately to the right of the call arrow segment, as shown in Figure 6.

More than one call reference may be attached to one call arrow. Should there be more than one call reference, the call references shall be left-justified within a vertical list. The last or bottom call reference in such a list shall be shown in exactly the same position relative to a call arrow's arrowhead as a single call reference. The order of call references in such a list has no significance. A calling box may not have a box detail reference because a calling box is not detailed by its own child diagram. A call reference may not be made to any kind of context diagram because a called diagram must be a decomposition diagram. For example, a calling box in a high-level context diagram for one model may incorporate another whole model only by invoking that model's ***A0*** diagram, which is the highest decomposition diagram in any IDEF0 model.

Different calling boxes may invoke the same called diagram and different called diagrams may be invoked by the same calling box. However, a calling box may invoke only one called diagram in any given activation of the calling box. Which activation conditions determine which called diagram is to be invoked by a calling box shall be specified in an accompanying model note. This note may refer the reader to a text page by a page reference for a complete explanation.

The boundary arrows of a called diagram need not correspond exactly with those of the calling box, either in number, name, or meaning. In these cases, a model note in the diagram containing the calling box shall specify these relationships so that a reader may appropriately interpret the shared physical and data objects required for an activation of the called diagram. This note may refer the reader to a text page by a page reference for a complete explanation.

6. IDEF0 semantics

Semantics involves, first, the interpretation of the basic syntactic components of a language and, second, how the interpretations of those basic components contribute to the meanings of more complex syntactic structures. Thus, a semantics for IDEF0 specifies how the boxes and arrows are to be interpreted and, in turn, how they contribute to the meanings of more complex structures built up from them.

6.1 Box/arrow semantics

a) *Box semantics.* A box models a *function*, i.e., an activity that, typically, takes certain inputs and, by means of some mechanism, and subject to certain controls, transforms its input into output. Thus, a *function name* shall be an active verb or verb phrase, such as "Inspect Parts," that describes what is accomplished by the function that the box represents. The name of a box is the name of the function represented by the box. Thus, for instance, a function named "Inspect Parts" would be

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

expected to transform uninspected parts into inspected parts, and the name of the box representing this function would also be "Inspect Parts."

An IDEF0 transformation may consist of any combination of these four kinds of transform: form, state, time, and place.

1) A transformation of *form* substantively changes input objects into different output objects.

2) A transformation of *state* changes some characteristic(s) of input objects to produce the output objects. IDEF0 modeling treats something in one state before an IDEF0 transformation and the same thing in another state after the transformation as distinctly different objects.

3) A transformation of *time* changes the temporal location of input objects, i.e., moves input objects at one point in time to be output objects at another point in time, without necessarily changing their form or state. A transformation of time is the IDEF0 expression of storage.

4) A transformation of *place* changes the spatial location of input objects, i.e., moves input objects at one location to be output objects at another location, without necessarily changing their form or state. A transformation of place is the IDEF0 expression of movement or flow.

b) *Output without input.* An IDEF0 function must be provided control and must provide output. Input is not required for a syntactically correct statement of an IDEF0 activity (see 5.4). From the particular viewpoint and for the specific purpose of a given model, IDEF0 input for an activity may be omitted if all inputs meet one or more of these criteria:

1) *Control-bundled input*. An input is bundled with a control (see 6.6).

2) *Nonobject input*. An input is not a physical or data object. Not everything that is real can be modeled as an IDEF0 object. Real things such as emotions, knowledge, political power, and planetary orbits exist but cannot be represented by IDEF0 arrows.

3) *Nonconstraint input*. An input is always available to a function in all circumstances relevant to a given model. Such input—a precondition assumed always to be true—does not constrain the function.

4) *Creative act.* The function itself represents a creative act, a transformation that essentially creates something from nothing. As creative acts, functions such as *Write Program* and *Compose Sonata* are typically modeled as activities whose outputs are created without transformation of IDEF0 input.

c) *Arrow semantics.* At some level of abstraction, an IDEF0 arrow models physical and data objects that may exist in the tangible world and an arrow label identifies the kinds of objects that are represented by an arrow. However, because IDEF0 arrow segments may branch and join to commingle arrow meaning in arbitrary ways, distinctions among object type sets, object types, object sets, and objects themselves are necessary to allow an arrow segment to express the absence as well as the presence of objects and to allow consistent interpretation of branches and joins (see 6.2).

1) An *object type set* is a named set of one or more object types. An object type set may include object types that are themselves grouped as object type sets. An object type set is designated by an arrow label.

2) An *object type* is the set of all possible instantiations of a singular concept, either physical or data, within an IDEF0 model.

3) An *object set* is a subset of instantiations from the set of all possible instantiations of all *object types* within a named *object type set*. An object set is a subset of the union of the members of an object type set; the set of object sets includes the null or empty set and the set of the union of the members of the object type set. An object set is modeled by an arrow segment. Object sets are not explicitly named in IDEF0 syntax.

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

4) An *object* is one instantiation of an object type, that is, a member of an object set. A physical or data object represents something in the observable world that may be distinguished from other instances of its object type and may be uniquely named. However, an object is not explicitly named in IDEF0 syntax.

Each arrow segment shall be identified by the object type set of the object set modeled by the arrow segment. Generally, however, the terms "*an arrow represents an object type set*" or "*an arrow represents physical and data objects*" are used to express the relationship between an object type set, which is designated by an arrow label, and an object set, which is modeled by an arrow segment.

d) *Box/arrow attachments.* An arrow segment that is attached to a box represents an object type set in a specific role—input, control, mechanism, or output—for the function that the box represents. The side of a box to which an arrow is attached defines the role of an object type set with respect to the function represented by that box. (As noted in 5.5, call arrows play no semantic role but are merely pointers to diagram pages that may detail the calling box.)

1) *Input.* An arrow attached to the left side of a box represents input for the function. Such an input arrow represents what is transformed or consumed by the function to produce the function's output.

2) *Control.* An arrow attached to the top of a box represents control for the function. Such a control arrow represents conditions that must be met before the function can produce correct output.

3) *Output.* An arrow attached to the right side of a box represents output from the function. Such an output arrow represents what is produced by the function.

4) *Mechanism.* An arrow attached to the bottom of a box represents mechanism for the function. Such a mechanism arrow represents the means that carry out the function.

Input to an activity, i.e., an object set modeled by an arrow segment attached to the left side of a box, must be transformed by some activation of the function into some output of the activity, i.e., a different object set modeled by an arrow segment attached to the right side of that box. Output may result from transformation of inputs specified for an activity; output must account for all input and all input must be accounted for by output. Neither control nor mechanism may be transformed by a function. Thus, all input, control, output, and mechanism arrows represent object type sets, but only a control or mechanism arrow may represent an object type set that contains only one possible instantiation.

Arrows do *not* represent temporal sequencing as in a traditional process or data flow model; indeed, the concept of time cannot be formally expressed through IDEF0 syntax. Rather, arrows represent object type sets. These object type sets are necessary to specify a transformation represented by a box. The role that an object type set plays in this transformation is specified by attaching the appropriately labeled arrow to the appropriate side of the box. That one box is connected to another by an arrow simply specifies that the output of one function provides input, control, or mechanism needed for the activation of the other function.

Standard arrow attachments are shown in Figure 6.

6.2 Branching and joining arrows

Three or more arrow segments may connect at one point to form a *junction*. Such a junction is either a branch or a join. A *branch* splits one arrow segment, the root segment, into two or more arrow segments, the branching segments. A *join* combines two or more arrow segments, the joining segments, into a single arrow segment, their root segment. Branches and joins may be interpreted symmetrically with respect to their root segments. In both kinds of junction, the meaning of the root arrow segment—that is, the object

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

types represented by the root segment, without regard to how these object types may be grouped or otherwise bundled as an object type set—shall correspond to the object type sets represented by the arrow segments that connect to it.

Specifically, the meaning of the root segment of a join shall be equivalent to the *union* of the meanings of all arrow segments that join it. The meaning of a root segment of a branch shall be equivalent to the *union* of the meanings of all arrow segments that branch from it. The *union* of the meanings of all segments that join a root segment shall be equivalent to the *union* of the meanings of all segments that branch from the same root segment. Thus, additionally, the meaning of a root segment shall be equivalent to the *union* of the meanings of all arrow segments that connect to it. In other words, the *set* of arrow segments that connect in a junction to a root arrow segment must represent *all of and nothing but* the meaning of that root arrow segment.

This conservation of meaning of object types through arrow branching and joining applies to the *composite* meaning of a complete *set* of joining segments, the *bundled* meaning of their root segment, and the *composite* meaning of a complete *set* of segments branching from that root. This conservation of meaning of object types does not apply to the meaning of individual arrow segments that connect to a junction as joining or branching segments. While a *set* of connecting arrow segments, branching or joining, must represent *all of and nothing but* the meaning of the root segment of a junction, this conservation of meaning does not require that different object types of a root's meaning be apportioned among the segments that connect to the root segment in any particular way. This conservation of meaning *does* require each branching or joining arrow segment to represent a subset of the union of the members of the object type set represented by the root arrow segment of a junction.[2]

In general, the object type set of any individual arrow segment that connects to a root segment at a junction may contain any arbitrary combination of object types that are also contained within the meaning of the root segment, regardless of the meaning of any other segments that may also connect to the same root segment. However, two special cases frequently appear in IDEF0 models: fungible junctions and abstract junctions. These variations are illustrated in Figure 10.

a) In a *fungible junction*, exactly the same object type set is represented by every arrow segment that connects to the root segment. In this case, the separate meanings of the branching or joining segments will be exactly the same as the meaning of the root segment of the junction; the meaning of each connecting segment replicates the meaning of the root segment.

b) In an *abstract junction*, the meaning of the root segment is the *set addition* of the meanings of the other connecting segments. In this case, the *set addition* equals the *set union* of the meanings of the arrow segments that connect to a root segment in a junction. An abstract join *bundles* differentiated groups of physical and data objects into a generalized or more abstract grouping construct: the component parts, which are represented by joining arrow segments, are *bundled* into the root arrow segment. Such bundling is also termed *arrow aggregation*. Conversely, an abstract branch *unbundles* a generalized or more abstract grouping construct into differentiated or less abstract object type sets: the root segment is *unbundled* into its component parts, which are represented by branching arrow segments. Such unbundling is also termed *arrow decomposition*.

   When a junction is abstract, either join or branch, the meanings of the segments that connect to the root segment are disjoint, i.e., there is neither redundancy nor overlap of meaning among these arrow segments. In particular, when an abstract branch unbundles a root segment formed by an abstract join, the separate meanings of the branching segments will be the same as the separate meanings of the root's joining segments; in other words, the set of arrow labels identifying joining

[2]This conservation of meaning also determines that an arrow segment in an IDEF0 model may not represent anything that is not specified as input, control, output, or mechanism for some function within the scope of the model and that the set of all arrow segments in an IDEF0 model must represent everything that is required as input, control, output, and mechanism for all functions within the scope of the model.

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

segments must be the same as the set of arrow labels identifying branching segments. Because an arrow label specifies what an arrow segment represents by identifying an object type set, arrow labels on branching (joining) arrow segments may detail the content of a root arrow segment just as decomposition diagrams detail a parent box.

For example, given a root arrow segment that models *apples*, *oranges*, and *mangoes* as a bundle of *fruit*, any set of arrow segments that join into *fruit* or any set of arrow segments that branch from *fruit* must represent, in any combination, *apples* and *oranges* and *mangoes*; further, neither *bananas* nor *apricots* may be represented by any segment joining to or branching from this particular bundle of *fruit*.

An arrow segment by itself models an object set—a specific set of instantiations of the object types identified by the label attached to the arrow segment. The IDEF0 box/arrow syntax models such an object set in a specific role with respect to a given activity. In particular, any attachment of an arrow to a box as an input, control, or mechanism expresses an *activation constraint*, that is, a requirement for the existence of a nonempty object set in a particular role before a function can execute (see 6.7).

An arrow segment models an object set without enumerating or counting the members of the object set. Just as time is not formally represented by an IDEF0 model, the idea of quantity also cannot be formally represented through IDEF0 syntax. An arrow segment models objects in an object set without expressing any quantitative meaning.

Rather, the attachment of an input, control, or mechanism arrow to a box denotes that the function cannot activate unless sufficient instances of the arrow's object types (that is, an object set) are available to the function. What "sufficient" might mean is not determined in an IDEF0 model by characterizing an arrow. Whatever "sufficient" might mean in a counting sense, "sufficient" is expressed in an IDEF0 model by the structure of IDEF0 controls, which prevent activation of a function until sufficient resources of the appropriate kinds are available and which stimulate activation of other functions until these sufficient resources are produced.

Thus, in an IDEF0 model, objects are not quantified or counted; the significance of this absence of quantity is that conservation of arrow meaning through junctions, i.e., through joins and roots and branches, does not require or imply that an arrow segment represent any specific number of objects. While IDEF0 box/arrow syntax can be used to specify, say, that *apples*, *oranges*, and *mangoes* are required to activate a function, this syntax cannot be used to specify *how many* apples, *how many* oranges, or *how many* mangoes are required. Thus, conservation of arrow meaning does not imply that the *number* of objects represented by a set of segments branching from a root segment conserves the *number* of objects represented by a corresponding set of segments joining that root.

6.3 Arrow meaning conventions

Because the meanings of arrow segments connected in a junction may be ambiguous, IDEF0 provides several conventions to clarify these meanings. These conventions also minimize the number of arrow labels that need be physically provided in a diagram. The conventions and their exceptions are illustrated in Figures 12 through 21. The conventions are

a) The meaning of any boundary arrow, expressed by its ICOM label, may be propagated to all unidentified connecting arrow segments, both joining and branching, unless this propagation is halted by

   1) A *labeled segment* (i.e., an arrow segment that is explicitly identified by an arrow label or an ICOM label),

   2) An *ambiguous segment* (i.e., one arrow segment that can be implicitly identified by different arrow labels via label propagation along different arrows), or

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 31320-1:2012

3) *Ambiguous attachments* (i.e., different arrow segments, all attached to the same box, that can be implicitly identified by the same arrow label via label propagation along different arrows).

In particular, an ICOM label for an input, control, or mechanism boundary arrow may identify all unidentified branching arrow segments, including subsequent arrow segments of that boundary arrow and arrow segments that branch from it. In addition, an ICOM label for an output boundary arrow may identify all unidentified joining arrow segments, including antecedent arrow segments in that boundary arrow and arrow segments that join it.

b) After ICOM labels of boundary arrow segments have been propagated, the arrow labels attached to other root arrow segments may be similarly propagated to branching arrow segments. The meaning of such root arrow segments, expressed either explicitly by an arrow label or implicitly by propagation of an ICOM label, may be propagated through all connected arrow segments, unless, as with boundary arrow ICOM labels, such propagation is halted by a labeled segment, an ambiguous segment, or ambiguous attachments.

Only ICOM labels and then arrow labels attached to the root segment of a branch may be propagated. In particular, an explicit arrow label attached to an arrow segment formed by joining arrow segments may not be propagated to those joining arrow segments. In contrast, if such a root segment may be implicitly identified by propagation of an ICOM label, the implicit identity of the bundled root segment may propagate to other, unidentified joining arrow segments.

6.4 Ambiguous arrow segments

The meaning of an arrow segment may propagate to another arrow segment only if it is the only meaning that may be propagated to that arrow segment. An ambiguous arrow segment arises if the meaning of the same arrow segment can be differently implied by the propagation of different labels along different arrows. An ambiguous arrow segment shall be explicitly labeled and may not be implied. A label may neither be propagated to nor through an ambiguous segment. In particular,

a) The meaning of an arrow segment may not propagate to an unidentified arrow segment that connects two arrow segments by branching from one and joining to the other, whether labeled explicitly or implicitly by propagation.

b) The meaning of an arrow segment may not propagate to an unidentified arrow segment that results from the join of arrow segments that have different meanings. Similarly, the meaning of an arrow segment may not propagate to an unidentified arrow segment that is the root of branching arrow segments that have different meanings, whether labeled explicitly or implicitly by propagation.

6.5 Ambiguous arrow attachments

The meaning of an arrow segment may propagate to only one arrow segment attached to a given box. Ambiguous arrow attachments arise if the meaning of an arrow segment can be propagated along different arrows to multiple arrow segments attached to the same box. An arrow segment with an ambiguous arrow attachment shall be explicitly labeled and may not be implied. A meaning may not be propagated to an arrow segment with an ambiguous attachment. In particular,

a) The meaning of an arrow segment may not propagate to any output arrow segment attached to a given box if that meaning can also propagate to any input, control, or mechanism arrow segment attached to the same box. An arrow segment meaning may propagate to an output arrow segment if and only if that arrow segment label cannot also propagate to any input, control, or mechanism arrow segment attached to the same box.

b) The meaning of an arrow segment may not propagate to any output arrow segment attached to a given box if that meaning can also propagate to any other output arrow segment attached to the same box. An arrow segment meaning may propagate to an output arrow segment attached to a

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

given box if and only if that arrow segment label cannot also propagate to any other output arrow segment.

c) The meaning of an arrow segment may not propagate to any input, control, or mechanism arrow segment attached to a given box if that meaning can also propagate to any other input, control, or mechanism arrow segment attached to the same box. An arrow segment meaning may propagate to an input, control, or mechanism arrow segment attached to a given box if and only if that arrow segment label cannot also propagate to any other input, control, or mechanism arrow segment.

6.6 Arrow role conventions

A significant IDEF0 graphical convention allows a single control arrow to represent both control *and* input for the same box if these arrow segments both branch from the same root. In this exceptional convention, roles rather than objects are bundled in the control arrow. This convention reduces the complexity of a diagram and emphasizes the control role of the bundled arrow; these features are displayed in Figure 11. An arrow attached to an ancestral box as a control can appear in a descendent diagram as a control, as an input, or as both a control and an input, depending on the relationship of the bundled arrow to the activities modeled in the descendent diagram. When this convention is used, the detail of separate input and control roles shall be shown in a descendent diagram. The diagram exhibiting this detail shall show the unbundling of these arrow roles into separate inputs and controls and their attachments to different boxes for different purposes.

An object may have more than one role with respect to an activity; the role is not inherent in the nature of the object. This convention allows two roles to be represented by one arrow for convenience and to reduce diagram clutter. This convention does *not* change control semantics into input semantics: an object in the role of control may *not* be transformed into output, while an object in the role of input *must* be transformed into output.

6.7 Activations

Models are general descriptions: they describe the general structure of a given process, the pattern exhibited by any given instance, or *activation*, of that process. It is very important to note that there need not be a contribution from every input, control, and mechanism arrow attached to a box in every activation of the function represented by the box, nor need there be output of the sort indicated by every output arrow. Under different circumstances, a box may perform selected parts of its function using different combinations of input, control, and mechanism to produce different output. Hence, in general, a diagram represents many different possible activations of any modeled process.

In particular, any attachment of an arrow to a box as input, control, or mechanism expresses an *activation constraint*, that is, a requirement for a nonempty object set in a particular role as a precondition to some execution of the activity; the function cannot act to produce some output in the absence of objects specified by the nonempty object set. Which activations of a function are so constrained is determined by the decomposition diagrams that detail the box to which the activation constraint applies.

6.8 Concurrent activation

Several functions in a model may be performed concurrently, if the needed constraints have been satisfied. As illustrated in Figure 22, output of one box may provide some or all the physical and data objects needed for activations of one or more other boxes. When output of one box provides some or all the input, control, or mechanism needed by another box, a given activation of the latter box may depend on sequential performance.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

7. IDEF0 diagrams

Several types of diagrams exist within an IDEF0 model. This clause identifies and discusses the types of IDEF0 diagrams and their relationships.

The graphic IDEF0 *diagram* is the primary component of an IDEF0 model. An IDEF0 diagram contains named boxes and labeled arrows; a diagram structures these graphic elements in accordance with the syntactic and semantic rules of IDEF0. A box represents a function (see 6.1). A function may be analyzed or *decomposed* into a more detailed diagram, and the boxes in such a decomposition diagram may in turn be further decomposed into diagrams providing additional detail. Such decomposition continues until a subject is rendered at the level of detailed exposition necessary to support the stated purpose of a particular model.

The starting point for this analysis is the ***A-0*** *context diagram*, which provides the most general or abstract description of the subject represented by a model. The ***A-0*** context diagram introduces a series of one or more *decomposition diagrams*, starting with the ***A0*** diagram, which provide successively deeper analysis of the modeled subject.

The ***A-0*** context diagram also defines the immediate context of the ***A0*** function of an IDEF0 model; *optional context diagrams* may provide further environmental context to expand and augment this immediate context. These optional context diagrams include the ***A-1*** *context diagram*.

All decomposition diagrams in an IDEF0 model *descend* from the model's ***A-0*** context diagram. Conversely, all context diagrams in an IDEF0 model *arise* from the model's ***A0*** decomposition diagram.

7.1 Diagram identification

A diagram is identified within an IDEF0 model by both a unique diagram number (see 10.3) and a unique diagram title. The diagram number and the diagram title that identify a diagram shall be recorded on the diagram page that contains the identified diagram. With the exception of the ***A-0*** context diagram and of the highest optional context diagram, the diagram title shall be the same as the box name of the diagram's parent box and the diagram number shall be the same as the node number of the diagram's parent function.

Neither the ***A-0*** context diagram nor the highest optional context diagram has a parent box from which to inherit its diagram title nor a parent function from which to inherit its diagram number. A diagram title for these diagrams shall combine the phrase "Context of" with a substantive equivalent of the box name of the most important box in the diagram. For example, given an ***A-0*** context diagram whose single box contains the box name "Build Ship," the diagram title of this context diagram would be "Context of Ship Building." In contrast, the diagram numbers for these context diagrams are a part of the IDEF0 language itself (see 10.3).

7.2 A-0 context diagram

An IDEF0 model shall include an ***A-0*** context diagram (see Clause 4) to be parent to the model's ***A0*** decomposition diagram. This diagram identifies the model, determines the subject of the model, and defines the scope of analysis to be included in the model. This diagram also defines the model's interfaces with other activities that are outside the scope of the model. This context diagram is always called the "A-0 context diagram," where "A-0" is pronounced "A minus zero," regardless of the actual node letter used in a model (see 10.2).

The *only* function shown in an ***A-0*** context diagram is the ***A0*** function of a model. In the ***A-0*** context diagram of a model, the ***A0*** function of the model is represented by box ***0***. This ***A0*** function represents the whole of the subject of the model; it is the unique parent of the entire modeled subject and thus the

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

ancestor of all activities modeled. By representing the ***A0*** function, the ***A-0*** context diagram sets a model's scope, that is, the boundaries of what may be included in that model.

The necessary and sufficient context required to understand and to analyze the ***A0*** function according to the model's viewpoint and purpose is given by boundary arrows that identify inputs, controls, and mechanisms required by the ***A0*** function and outputs that it produces. The inputs, controls, and mechanisms for the ***A0*** function are made available by activities that are outside the scope of a model. Similarly, the outputs of the ***A0*** function are used by activities that also are outside the scope of a model. These external activities shall not be analyzed or represented in the decomposition diagrams of a model. Because the context for the ***A0*** function as given by the ***A-0*** diagram does not identify these external functions, this context is often called the *immediate* context of the ***A0*** function.

All decomposition diagrams are strictly bound in scope to the limits of the ***A0*** function as specified by the ***A-0*** context diagram. In particular, the boundaries of a model's ***A0*** decomposition diagram are precisely coextensive with the boundaries of the model's ***A-0*** context diagram. Thus, an ***A-0*** context diagram is the strict parent of its ***A0*** decomposition diagram and the ancestor of all further decomposition diagrams.

The ***A-0*** context diagram of an IDEF0 model shall contain only one box, box ***0***, which represents the ***A0*** function of the model. The ***A-0*** context diagram shall present the model name, the model name abbreviation, and brief text statements that specify the viewpoint and the purpose of the model; these statements will guide and constrain the model's development. Boundary arrows connected to box ***0*** shall identify the necessary and sufficient inputs, controls, mechanisms, and outputs required to establish the immediate context of the ***A0*** function. These boundary arrows may not have ICOM codes or tunnel notations at their unconnected ends.

All decomposition diagrams in an IDEF0 model *descend* from the model's ***A-0*** context diagram. Conversely, all context diagrams in an IDEF0 model *arise* from the model's ***A0*** decomposition diagram. Due to its unique responsibility to define the purpose and scope of an IDEF0 model, an ***A-0*** context diagram does not itself have a parent diagram; instead, the parental context of the ***A-0*** diagram is defined by the IDEF0 language to be the token **TOP** (see also 7.7).

An example ***A-0*** context diagram is shown in Figure 23.

7.3 Model name

Each model shall be given a unique, descriptive *model name* that distinguishes it from other models. A unique *model name abbreviation* shall be derived from this model name; the model name abbreviation is used to construct diagram reference expressions.

A model's full name and its abbreviation shall be placed together in the ***A-0*** context diagram of the IDEF0 model. A model name abbreviation shall consist of two or three uppercase alphanumeric characters. For example, a model named "Manufacturing Operations" may be abbreviated **MFG** (see 10.6 for a discussion of diagram references). A model name abbreviation is typically given in parentheses following the model name, in this way: "Maritime Pipe Laying (MPL)." A model name is a required textual element in an ***A-0*** context diagram, and, therefore, shall not be marked as a model note. Each model shall have only one model name and one model name abbreviation.

7.4 Model viewpoint

A *viewpoint statement* guides and constrains a model's development. This statement determines what can be seen within a model's context and shapes what a model will notice and pay attention to. Different viewpoints emphasize different aspects of the same subject and necessitate different models. Functions, objects, and data that are important in one viewpoint may not even appear in a model developed from another viewpoint of the same subject.

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

The viewpoint statement for a model shall be placed in the ***A-0*** context diagram of the model and is a brief sentence that identifies a person or a personified role. The perspective of this person or role determines the model's viewpoint. A viewpoint statement is a required textual element in an ***A-0*** context diagram, and, therefore, shall not be marked as a model note. Each model shall have only one viewpoint statement.

7.5 Model purpose

A *purpose statement* guides and constrains a model's development and expresses the reason a model is created. The purpose statement determines the question that is to be answered by a model. When the purpose statement is fulfilled by a satisfactory answer to the question posed, the modeling effort concludes.

The purpose statement for a model shall be placed in the ***A-0*** context diagram of the model and is a brief sentence that identifies the question addressed by a model. A purpose statement is a required textual element in an ***A-0*** context diagram, and, therefore, shall not be marked as a model note. Each model shall have only one purpose statement.

7.6 Optional context diagrams

An IDEF0 model may include context diagrams (see 7.2) that provide a fuller exposition of the environment of a model. These optional diagrams examine the environment of the ***A0*** function and model a *broader* scope than the ***A-0*** context diagram. Because optional context diagrams incorporate functions that are outside the scope of a model's decomposition diagrams, these diagrams are often called the *environmental* context of the ***A0*** function.

By providing a more extensive description of a modeled system's environment, optional context diagrams provide more constraining specifications on the boundary conditions of a modeled system. Contextual modeling provides details about the sources and uses of the physical and data objects modeled as boundary arrows in the ***A-0*** context diagram. Such detail may not precisely match any particular "real world" environment but this detail is not intended to be definitive. Optional context diagrams serve their purpose sufficiently by describing typical environmental contexts.

Optional context diagrams add functions to the environment of the ***A0*** function of a model. These functions are outside the scope of the ***A0*** diagram as established by the ***A-0*** context diagram. The boundary arrows of the ***A-0*** context diagram specify the inputs, controls, and mechanisms required by the ***A0*** function and the outputs it produces. These neighboring functions provide the ***A0*** function with these inputs, controls, and mechanisms. These functions can also use outputs of the ***A0*** function for their own activations. Optional context diagrams illuminate this set of mutually constraining dependencies and so define an environment that is necessary for the ***A0*** function to operate.

— *The A-1 context diagram.* The first of the optional context diagrams is the ***A-1*** context diagram. The ***A-1*** context diagram (pronounced "A minus one") is constructed to be a second parent to a model's ***A0*** decomposition diagram (see also 7.2). This parent/child relationship between the ***A-1*** and the ***A0*** diagrams links the optional context diagrams to the decomposition diagrams that descend from the ***A-0*** context diagram. Because the ***A0*** function appears in ***both*** the ***A-0*** and ***A-1*** context diagrams, ***both*** context diagrams are parents of the model's ***A0*** diagram. If present, the ***A-1*** context diagram replaces the ***A-0*** context diagram as the parental context of the ***A0*** diagram.

— *Lineal ancestry.* Optional context diagrams include *high-level* context diagrams and their contextual descendants. High-level context diagrams are lineal ancestors of the ***A0*** diagram because they decompose the lineal ancestors of the ***A0*** function. Their contextual descendants decompose other functions that are not lineal ancestors of the ***A0*** function. The *high-level* context diagrams ***A-1***, ***A-2***, ***A-3***, ***A-4***, ***A-5***, ***A-6***, ***A-7***, ***A-8***, and ***A-9*** are defined as the *lineal ancestors* of the

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

A0 diagram. The ***A-1***, ***A-2***, ***A-3***, ***A-4***, ***A-5***, ***A-6***, ***A-7***, and ***A-8*** functions are defined as the *lineal ancestors* of the ***A0*** function.

The ***A-1*** context diagram is one parent of the ***A0*** diagram (the other parent is the ***A-0*** context diagram). The grandparent of the ***A0*** diagram is the ***A-2*** context diagram, the great-grandparent of the ***A0*** diagram is the ***A-3*** context diagram, and so on, through the ***A-9*** context diagram, which is the highest ancestry allowed. (An ***A-1*** context diagram is *not* parent to the ***A-0*** context diagram.)

A model may contain up to nine lineal ancestors to the ***A0*** diagram as optional context diagrams. The last ancestral diagram in a sequence of lineal ancestors will have no parent diagram and is called the *highest* optional context diagram. All parent/child diagram relationships between the highest optional context diagram and the ***A0*** diagram must be included in a model. Thus, if the highest optional context diagram is the ***A-5*** context diagram, then the ***A-1***, ***A-2***, ***A-3***, ***A-4***, and ***A-5*** context diagrams must all be included in the model.

Figure 25 illustrates, in node-tree form, the diagram structure of a possible environmental context for the ***A0*** function of an IDEF0 model. Both the ***A-0*** and the ***A-1*** context diagrams are parents to the ***A0*** diagram. The ***A-2*** and ***A-1*** context diagrams are lineal ancestors of the ***A0*** decomposition diagram. The ***A-2*** diagram is the grandparent of the ***A0*** diagram and the highest context diagram in this model. The ***A-24*** and ***A-212*** context diagrams are contextual descendants of the lineal ancestor ***A-2***. The ***A-13*** and ***A-141*** context diagrams are contextual descendants of the lineal ancestor ***A-1***. Note that all context diagrams are characterized by negative diagram numbers. Note also that the ***A-2*** diagram is parent to diagrams ***A-21***, ***A-22***, ***A-1***, and ***A-24***. By the rules that associate box numbers, node numbers, and diagram numbers, there must be boxes with box numbers ***1***, ***2***, ***-1***, and ***4*** in diagram ***A-2***. Diagrams ***A-21***, ***A-22***, and ***A-24*** are contextual descendants of the ***A-2*** diagram while diagram ***A-1*** is a lineal ancestor of the ***A0*** diagram.

A model may contain any number of optional context diagrams; functions shown in optional context diagrams may be decomposed in the ordinary way (see 7.7). A function that provides or uses physical and data objects modeled as boundary arrows in the ***A-0*** diagram may be represented by a box in any optional context diagram. The environmental context is not required to identify functions to provide every input, control, or mechanism or to use every output defined in the ***A-0*** diagram. A boundary arrow in the ***A-0*** diagram may also be represented as a boundary arrow in the highest optional context diagram, and such representation indicates that the source or use of this physical or data object is not examined within the presented environmental context.

Optional context diagrams shall follow the same syntactic and semantic rules as decomposition diagrams, with these exceptions:

a) Diagrams, functions, and boxes that represent the lineal ancestry of the ***A0*** function shall be numbered using predefined negative numbers (see Clause 10).

b) In each optional context diagram that is a lineal ancestor of the ***A0*** diagram, one box shall be assigned a negative box number.

c) A function that is a lineal ancestor of the ***A0*** function, i.e., a function identified in an optional context diagram by a predefined negative node number, shall be decomposed.

d) A box with a negative box number represents a function that is a lineal ancestor of the ***A0*** function. Such a box shall have a box detail reference.

e) Arrows shall not be tunneled in optional context diagrams. Each arrow attached to a parent box shall correspond to a boundary arrow on the child diagram. Arrow tunneling is not permitted in any optional context diagrams, except to show tunnel notation that may have been used in the ***A-0*** diagram.

Additional requirements for the ***A-1*** context diagram are

Copyright © 1998 IEEE. All rights reserved.

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012

— The ***A-1*** context diagram shall contain one box with the box number ***0*** to represent the ***A0*** function of a model (see 10.1).

— The immediate context of the ***A0*** function as specified in the ***A-0*** diagram shall be repeated in the ***A-1*** diagram, that is, all physical and data objects modeled by boundary arrows attached to box ***0*** in the ***A-0*** context diagram shall be also be represented by arrows attached to box ***0*** in the ***A-1*** context diagram. Only those arrows shown in the ***A-0*** context diagram may be attached to box ***0*** in the ***A-1*** context diagram; no additional arrows may be attached to box ***0*** in the ***A-1*** diagram.

An example ***A-1*** context diagram is shown in Figure 24.

Additional requirements for the highest optional context diagram are

— The parental context of the highest optional context diagram is defined by the IDEF0 language as the token **NONE**.

— As with the ***A-0*** diagram, boundary arrows in the highest optional context diagram may not attach to ICOM codes.

7.7 Decomposition diagrams

Decomposition is the partitioning of a modeled function into its component functions. A *decomposition diagram* is a diagram that shows the detail of boxes and arrows for such a decomposition. The single function represented in the ***A-0*** context diagram is decomposed into its major subfunctions by creating its decomposition diagram. In turn, each of these subfunctions may be decomposed, each creating another lower-level decomposition diagram. A decomposition diagram is also called a *child diagram,* and the diagram that contains the box that is detailed by the decomposition is also called a *parent diagram*. Some of the functions, none of the functions, or all of the functions in a given diagram may be decomposed. Each decomposition diagram contains the boxes and arrows that provide additional detail about the decomposed box.

The diagram that results from the decomposition of a function covers the same scope as the box it details. Thus, a decomposition or child diagram may be thought of as the interior of the parent box it details. This decomposition structure is illustrated in Figure 26.

7.8 Parent/child diagram relations

A *parent diagram* is a diagram that contains one or more *parent boxes*, i.e., a box that is detailed by a decomposition diagram. Every ordinary (noncontext) 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 (a box detailed by a child diagram) and a *child box* (a box that appears in a child diagram). The primary hierarchical relationship is between a parent box and the child diagram that details it.

Thus ***A0*** is always the node number of the top-level function in a model and at the same time the diagram number of its child diagram; together, these represent the whole model. The model's parent ***A0*** function is always detailed by functions with node numbers ***A1***, ***A2***, ***A3***, to at most ***A9***.

The parental context provides or names the context for a diagram without its own parent diagram. If there is no ***A-1*** context diagram, the parental context of the ***A0*** diagram is the ***A-0*** context diagram. If there is an ***A-1*** context diagram, the ***A-1*** diagram is taken to be the parent of the ***A0*** diagram.

A box that is not further decomposed is called a *leaf node* of the decomposition. A leaf node is represented as a box that does not have the enclosed corner known as a box detail reference.

Copyright © 1998 IEEE. All rights reserved.
AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 31320-1:2012