INTERNATIONAL STANDARD

ISO/IEC/ IEEE 31320-2

First edition
2012-09-15

Information technology — Modeling Languages —

Part 2: Syntax and Semantics for IDEF1X$_{97}$ (IDEF$_{object}$)

Technologies de l'information — Langages de modélisation —

Partie 2: Syntaxe et sémantique pour IDEF1X$_{97}$ (IDEF$_{object}$)

Reference number
ISO/IEC/IEEE 31320-2:2012(E)

COPYRIGHT PROTECTED DOCUMENT

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

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 31320-2 was prepared by the Software & Systems Engineering Standards Committee of the IEEE Computer Society (as IEEE 1320.2-1998). 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 Std 1320.2-1998(R2004)

IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X$_{97}$ (*IDEF*$_{object}$)

Sponsor
Software Engineering Standards Committee
of the
IEEE Computer Society

Reaffirmed 25 March 2004
Approved Approved 25 June 1998
IEEE-SA Standards Board

Abstract: IDEF1X$_{97}$ consists of two conceptual modeling languages. The key-style language supports data/information modeling and is downward compatible with the US government's 1993 standard, FIPS PUB 184. The identity-style language is based on the object model with declarative rules and constraints. IDEF1X$_{97}$ identity style includes constructs for the distinct but related components of object abstraction: interface, requests, and realization; utilizes graphics to state the interface; and defines a declarative, directly executable Rule and Constraint Language for requests and realizations. IDEF1X$_{97}$ conceptual modeling supports implementation by relational databases, extended relational databases, object databases, and object programming languages. IDEF1X$_{97}$ is formally defined in terms of first order logic. A procedure is given whereby any valid IDEF1X$_{97}$ model can be transformed into an equivalent theory in first order logic. That procedure is then applied to a meta model of IDEF1X$_{97}$ to define the valid set of IDEF1X$_{97}$ models.
Keywords: conceptual schema, data model, IDEF1X, IDEF1X$_{97}$, identity style, information model, key style, object model

---

The Institute of Electrical and Electronics Engineers, Inc.

345 East 47th Street, New York, NY 10017-2394, USA

Print ISBN 0-7381-0341-1 SH94663

PDF ISBN 0-7381-1405-7 SS94663

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.

Introduction

[This introduction is not a part of IEEE Std 1320.2-1998, IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X$_{97}$ *(IDEF$_{object}$)*.]

Background

The need for semantic models to represent conceptual schemas was recognized by the US Air Force in the mid 1970s as a result of the Integrated Computer Aided Manufacturing (ICAM) Program. The objective of this program was to increase manufacturing productivity through the systematic application of computer technology. The ICAM program identified a need for better analysis and communication techniques for people involved in improving manufacturing productivity. As a result, the ICAM program developed a series of techniques known as the ICAM Definition (IDEF) methods, which included the following:

a) IDEF0, a technique used to produce a "function model," which is a structured representation of the activities or processes within the environment or system.
b) IDEF1, a technique used to produce an "information model," which represents the structure and semantics of information within the environment or system.
c) IDEF2, a technique used to produce a "dynamics model," which represents the time-varying behavioral characteristics of the environment or system.

IDEF0 and IDEF1X (the successor to IDEF1) continue to be used extensively in various government and industry settings. IDEF2 is no longer used to any significant extent.

The initial approach to IDEF information modeling (IDEF1) was published by the ICAM program in 1981, based on current research and industry needs [B23].[1] The theoretical roots for this approach stemmed from the early work of Dr. E. F. Codd on relational theory and Dr. P. P. S. Chen on the entity-relationship model. The initial IDEF1 technique was based on the work of Dr. R. R. Brown and Mr. T. L. Ramey of Hughes Aircraft and Mr. D. S. Coleman of D. Appleton Company, with critical review and influence by Mr. C. W. Bachman, Dr. P. P. S. Chen, Dr. M. A. Melkanoff, and Dr. G. M. Nijssen.

In 1983, the US Air Force initiated the Integrated Information Support System (I^2S^2) project under the ICAM program. The objective of this project was to provide the enabling technology to integrate a network of heterogeneous computer hardware and software both logically and physically. As a result of this project and industry experience, the need for an enhanced technique for information modeling was recognized.

Application within industry had led to the development in 1982 of a Logical Database Design Technique (LDDT) by R. G. Brown of the Database Design Group. The technique was also based on the relational model of Dr. E. F. Codd and the entity-relationship model of Dr. P. P. S. Chen, with the addition of the generalization concepts of J. M. Smith and D. C. P. Smith. LDDT provided multiple levels of models and a set of graphics for representing the conceptual view of information within an enterprise. It had a high degree of overlap with IDEF1 features, included additional semantic and graphical constructs, and addressed information modeling enhancement requirements that had been identified under the I^2S^2 program. Under the technical leadership of Dr. M. E. S. Loomis of D. Appleton Company, a substantial subset of LDDT was combined with the methodology of IDEF1 and published by the ICAM program in 1985 [B15]. This technique was called IDEF1 Extended or, simply, IDEF1X.

In December 1993, the US government released a Federal Information Processing Standard (FIPS) for IDEF1X. FIPS PUB 184 [B13] was based on the ICAM program description of IDEF1X and additional features originally included in LDDT. The FIPS clarified and corrected points in the ICAM publication, sepa-

[1]The numbers in brackets correspond to those of the bibliography items listed in Annex A.

rated language syntax and semantics definition from practice and use issues, and provided a formal first-order language definition of IDEF1X.

IEEE Std 1320.2-1998 continues the evolution of the IDEF1X language. It is driven by two needs. First, development of a national standard for the language makes the definition more accessible to organizations that do not follow US government standards and allows consideration and inclusion of features needed outside the US federal government sector. Second, the needs of the users of a standard change over time as system development techniques and available technology continue to evolve. Some users adopt new concepts earlier than others. To be valuable to the widest set of users, this standard needs to support a range of practices, from those supported by the FIPS to those that are emerging as future drivers of integration.

The change in the drivers of integration is being recognized by both government and private sector organizations. Integration involves not only data but the operations performed on that data. The emerging object modeling approaches seek to treat all activities as performed by collaborating objects that encompass both the data and the operations that can be performed against that data. There is increasing interest in these approaches in both the government and private sectors. Original work done for the National Institute of Standards and Technology (NIST) in 1994 and early 1995 by Robert G. Brown of the Database Design Group (DBDG) provides the basic elements required for a graceful evolution of IDEF1X toward full coverage of object modeling [B5].

The DBDG work analyzed the 1993 definition of IDEF1X and compared to it to the emerging consensus object model. The analysis showed that

— The concepts of the current IDEF1X were a subset of those of the object model,
— The current IDEF1X contained restrictions that are unnecessary in the object model, and
— The object model contains significant new concepts.

The work also showed that if the concepts of IDEF1X were more fully developed, the restrictions dropped, and the new concepts added, the result would be an upwardly compatible object modeling technique. The evolutionary features of IDEF1X described in this standard draw heavily from the DBDG work done for NIST.

Base documents

The following documents served as base documents for the parts of IEEE Std 1320.2-1998 indicated:

a) *From IDEF1X to IDEF$_{object}$*, 1995, by Robert G. Brown, The Database Design Group, Newport Beach, CA, is the base document for the Class and Responsibility clauses. Partial financial support was provided by the National Institute of Standards and Technology (NIST) [B5].
b) *IDEF1X$_{97}$ Rule and Constraint Language (RCL),* 1997, by Robert G. Brown, The Database Design Group, Newport Beach, CA, is the base document for the RCL clause. Partial financial support was provided by the Defense Information Systems Agency (DISA) [B6].
c) *IDEF1X$_{97}$ Formalization,* 1998, by Valdis Berzins, Naval Postgraduate School, Monterey, CA, and Robert G. Brown, The Database Design Group, Newport Beach, CA, is the base document for the Formalization clause. Partial financial support was provided by DISA and the Defense Modeling and Simulations Office (DMSO) [B7].

The IDEF1X approach

A principal objective of IDEF1X is to support integration. The “IDEF1X approach” to integration focuses on the capture, management, and use of a single semantic definition of the data resource referred to as a *conceptual schema*. The conceptual schema provides a single integrated definition of the concepts relevant to an enterprise, unbiased toward any particular application. The primary objective of this conceptual schema is to

provide a consistent definition of the meanings and interrelationship of concepts. This definition can then be used to integrate, share, and manage the integrity of the concepts. A conceptual schema must have three important characteristics:

— It must be consistent with the infrastructure of the business and be true across all application areas.
— It must be extendible, such that, new concepts can be defined without disruption to previously defined concepts.
— It must be transformable to both the required user views and to a variety of implementation environments.

IDEF1X is the semantic modeling technique described by IEEE Std 1320.2-1998. The IDEF1X technique was developed to meet the following requirements:

— Support the development of conceptual schemas.
— Be a coherent language.
— Be teachable.
— Be well-tested and proven.
— Be automatable.

Organization of this document

This document begins with an explanation of the scope and purpose of this version of the IDEF1X standard. Clause 1 also describes the evolution of the IDEF1X standard. It provides a context for understanding the approach and constructs presented in the rest of this standard.

Clause 2 identifies additional references that must be on hand and available to the reader of this standard for its implementation. Other documentation and related references that might be of interest to the reader or that were used in preparing this standard are included in the bibliography (see Annex A).

This document uses words in accordance with their definitions in the *Merriam-Webster's Collegiate Dictionary* [B26]. A definitions clause (see Clause 3) is provided for the convenience of those not already familiar with the terminology in question. It also contains any terminology that has specialized meaning in the context of this standard.

Clauses 4 through 6 along with 8 discuss the meaning (semantics) of each model construct that may be used within an IDEF1X model, as well as how they shall be put together to form a valid model (the syntax). Clause 7 provides a full description of the Rule and Constraint Language (RCL) specification language for an IDEF1X model.

Clause 4 introduces the language constructs of IDEF1X. The basic constructs of an IDEF1X model are

a) Things whose knowledge or behavior is relevant to the enterprise, represented by boxes;
b) Relationships between those things, represented by lines connecting the boxes;
c) Responsibilities of those things, stated as
    1) Knowledge and behavior properties, represented by names within the boxes,
    2) Realization of those responsibilities, expressed as sentences in a declarative language, and
    3) Rules, represented as constraints over property values.

These constructs are then described in detail in Clauses 5 and 6. Clause 8 discusses how the various constructs may be put together to form a model.

Two styles of IDEF1X modeling are described in this standard. Clauses 5 through 8 present the *identity style*, which extends the conceptual schema representation capabilities of IDEF1X. Identity-style

models describe the structural dimension of an object model and specify the collaborations among the objects. Identity-style models can be used in conjunction with dynamic modeling techniques such as those based on finite state machines.

Clause 9 describes the *key style*, which is backward-compatible with FIPS PUB 184 [B13]. This style may continue to be used to produce models that represent the structure and semantics of data within an enterprise, i.e., data (information) models.

In the process of producing FIPS PUB 184 [B13], the various graphical constructs of the IDEF1X language were formalized. In essence, these constructs had no more meaning than they had before, but they became more explicitly grounded than they had been. The formalization served to make obvious the fact that the graphical aspect of IDEF1X was not the language *per se* but only one external manifestation of it. Clause 10 presents the formalization of the IDEF1X language, revised to include the language features defined in IEEE Std 1320.2-1998. The formalization also provides a metamodel of IDEF1X. In addition to the metamodel diagram, the metamodel value classes and constraints are given. The reader may wish to use this model along with Annex D, which documents the built-in classes of the IDEF1X metamodel.

Additional normative and informative annexes provide convenient reference to supporting material:

— Annex A is a bibliography of relevant reference material.
— Annex B summarizes the differences and similarities between the version of IDEF1X documented in FIPS PUB 184 [B13] and this standard. The reader familiar with FIPS PUB 184 may wish to review this information before proceeding into the body of IEEE Std 1320.2-1998.
— Annex C presents a set of examples illustrating various aspects of identity-style modeling. These examples include the representation of two patterns from *Design* Patterns [B14], a business example that applies these patterns, some value class examples, and the translation of the TcCo model from FIPS PUB 184 [B13] into an initial identity-style model.
— Annex D documents the built-in classes of the IDEF1X language.

Throughout this standard, IEEE conventions for certain words are used:

— “Shall” means “required.” For example, point 5.1.2.1 a) says “A class shall be represented as a rectangle of the shape appropriate to the class.” This statement is interpreted as a mandatory requirement that a rectangle be the only acceptable way to represent a class.
— “Should” means “recommended.” For example, point 8.1.3.7 a) says “If the objective of the view is that it be internally consistent, it should be possible to demonstrate that a consistent set of instances exists.” This statement means that the presentation of a set of instances is highly recommended but not required for conformance.
— “May” means “permitted.” For example, point 5.2.3.6 c) says that “In a sample instance table, the instance identity label may be shown to the left of the row representing the instance.”
— “Can” means “is able to.” For example, 7.5.3 states that “The uniqueness conditions guarantee that a message can be resolved to at most one class responsibility.”

Reading the document

The $IDEF1X_{97}$ ($IDEF_{object}$) standard was developed to extend the practice of information modeling ($IDEF1X_{93}$) to object modeling. The readers of this standard can be broadly classified into at least at two distinct groups: management and technical. For each group, a different approach to the reading of this standard is recommended.

Management readers

This standard is written on a fairly technical plane. Managers may wish to focus on the concepts that will help them manage projects that employ this new standard. For example, modeling now will include operations on enterprise knowledge as well as rules that govern state changes. For this group of readers, Clause 1 should be read first as the key to the document. Clause 1 delimits the scope and defines the purpose of the document, providing a succinct discussion of the evolution of IDEF1X and pointing out the capabilities added in $IDEF1X_{97}$. As this clause points out, $IDEF1X_{97}$ is considered a transition language that preserves the information modeling investment, provides opportunities to simplify the data/process approach, and positions the organization to move forward.

Clause 4 should be read next. This clause provides a high level summary of the language concepts, constructs, and notation. The notation is not terribly significant to management; however, many of the concepts and constructs summarized in Clause 4 will lead a management reader to further discussions of concepts and constructs found in Clauses 5 through 8.

In the past at least two separate requirement specification languages had to be used (e.g. IDEF0, “Function Modeling” and $IDEF1X_{93}$, “Information Modeling” languages). The $IDEF1X_{97}$ identity-style language represents concepts in a more natural way by integrating data and process and by hiding implementation details that sometimes become a barrier to specifying the requirements. Hiding the implementation detail (encapsulation) simplifies the development and maintenance of databases and software.

Encapsulation is enabled by the concept that a class instance has responsibilities (see Clause 6) specified in two parts: interface and realization. By revealing only the interface specification (names, meanings, and signatures of responsibilities) to a client, $IDEF1X_{97}$ hides the complexity of realizations (the implementation detail) and their specified methods and representation properties. The realization is specified separately with the RCL so that database and software projects developed using $IDEF1X_{97}$ can focus on specifying the desired behavior and optimizing the messages requesting the services.

Managers should find the concept of modeling levels in Clause 8 of particular interest. Three technology-independent levels (survey, integration, and fully specified) and one technology-dependent level (implementation) are presented to help provide clear work product definition for management.

Clause 9 and Annex B will show management how older style information modeling ($IDEF1X_{93}$) can be supported and extended with features of the new object language.

One of the most powerful aspects of $IDEF1X_{97}$ models is that they are, with suitable automation support, directly executable to prove their correctness. Managers generally will not need to study the details, but should be aware of where to find them. The executable nature is enabled by the concepts discussed in detail in Clause 10, with a supporting RCL explained in detail in Clause 7.

Technical readers

Several groups will have primarily technical interests in the $IDEF1X_{97}$ standard.

- **Architects and Methodologists:** Readers in this group are often responsible for developing
  - The structure given to database and software components, their interrelationships, and the principles and guidelines governing their design and evolution over time (architecture) and
  - The routine procedures and practices used to produce precise, consistent, and repeatable deliverables at the end of each stage of the development process (methodology). Generally, this technical group uses modeling languages like $IDEF1X_{97}$ to develop architectures and methodologies to guide others in building consistently high quality products.

- **Data Modelers, Object Modelers, Database Designers, Software Engineers, and Other Practitioners:** Readers in this group are often responsible for defining and specifying requirements, designing and developing databases and software system solutions, and then testing and implementing those solutions as quickly and efficiently as possible. Generally, this technical group uses modeling languages like IDEF1X$_{97}$ to develop models, designs, and products to define and satisfy operational requirements with the highest quality databases and software at the lowest risk and cost of maintenance.
- **Commercial Software Vendors:** This group includes companies that build software products and tools to support the other technical groups. These software products and tools include
  1) Database management systems, including relational, object-relational, and object-oriented,
  2) Languages (procedural and object oriented),
  3) Computer aided software engineering (CASE) tools, and
  4) Data dictionary/repository systems.

Each of these technical groups will be more naturally satisfied by different reading patterns. Although Clauses 1 and 4 provide an overview, readers in these groups will be most interested in the detailed technical topical discussions in the major clauses (Clauses 5 through 10) and the annexes.

Data modelers and database designers, for example, may want to know how the new language differs from the earlier versions of IDEF1X (Clause 9 and Annex B) and, perhaps, how to begin the transition to object modeling and design. Object modelers will need to understand all features and capabilities of IDEF1X$_{97}$ identity-style modeling.

Clause 5 delineates the two types of IDEF1X$_{97}$ classes (state and value) and describes the use of generalization and relationship concepts. For the data modelers, understanding how it is possible to use value classes in place of domains will be of interest. Object technicians in all technical groups will be interested in the value class approach and in the generalization and relationship concepts: variants of these concepts exist in many currently available object modeling and design tools and in commercial software.

If a data modeler does not intend to develop object models, Clause 6 will not be of any significant interest. However, all other technical readers should carefully read Clause 6 to gain a core understanding of the object constructs and the extent of their usage by IDEF1X$_{97}$. Clause 6 introduces the concepts of responsibility, interface, realizations, requests, properties, attributes, participant properties, operations, constraints, and notes. All technical readers should study the concepts of view, view level, environment, glossary, and model presented in Clause 8.

Clause 9 is intended for data modelers who want to or must continue the practice of key-style modeling. Other technical readers will have little interest in this clause unless they support the older style practices or are planning transitions from that style of practice to object-oriented technology. In these cases, a thorough reading of Clause 9 could help with planning for changes to architectures, methodologies, and commercial software products and tools.

For all technical readers, Clause 10 and its companion Clause 7 will present the precise definition of the language. These clauses will be a key area of study for tool builders.

Contents

IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X$_{97}$ (*IDEF$_{object}$*)

1. Overview

This standard describes the semantics and syntax of IDEF1X, a language used to represent a conceptual schema. Two styles of IDEF1X model are described. The *key style* is used to produce information models that represent the structure and semantics of data within an enterprise and is backward-compatible with the US government's Federal Information Processing Standard (FIPS) PUB 184*, Integration Definition for Information Modeling (IDEF1X)* [B13].[1] The *identity style* is used to produce object models that represent the knowledge, behavior, and rules of the concepts within an enterprise. It can be used as a growth path for key-style models. The identity style can, with suitable automation support, be used to develop a model that is an executable prototype of the target object-oriented system.

1.1 Scope

This standard defines the semantics and syntax of IDEF1X. It does so by defining the valid constructs of the language and specifying how they can be combined to form a valid model.

IDEFIX takes the approach that an enterprise manages what it knows about (its knowledge). Such knowledge consists of awareness about enterprise-pertinent actions, facts, and the relationships among them. In order to maximize the utility of this knowledge, it must be codified in a manner that makes its interpretation consistent. Without this guidance, the knowledge is either not understood at all or, worse, misused to draw unsupported or inappropriate conclusions. The guide to the interpretation and use of the enterprise knowledge has three components:

a) A grammar that dictates the kinds of actions, facts, and relationships that the enterprise is interested in recording,
b) Operations that can be performed on/with this knowledge to produce usable information, and
c) Rules about recorded knowledge that help the enterprise weed out conflicting statements and rules that govern the state changes that recorded knowledge can undergo.

[1] The numbers in brackets correspond to those of the bibliography items listed in Annex A.

For example, the sentence “The chair sings the tree” is grammatically sound in English; there is a subject, a verb, and an object in the sentence. However, the sentence itself is not useful because it states something that is nonsensical. In a natural language, rules must be established that, for instance, indicate that the subject of the sentence must be capable of taking action, if the verb is an action, and of taking the particular action specified by the verb.

Such a guide to the interpretation and use of the enterprise knowledge is, itself, captured as a set of facts. This body of facts about facts, or metaknowledge, in turn needs a guide to *its* understanding and use. This goal, in a nutshell, is the scope of IDEF1X. As part of its semantics and syntax, IDEF1X establishes just what can be said about the enterprise knowledge and what sorts of conclusions can be drawn from that meta-knowledge.

This standard does not treat methodology. A methodology is an ordered process used to produce a repeatable result. An IDEF1X methodology deals with the process of creating a model using the IDEF1X language. While critical to the practitioner, such considerations are beyond the scope of this standard. Rather, the IDEF1X constructs will be presented individually, without regard for their logical sequence of use.

1.2 Purpose

This purpose of this standard is to describe the IDEF1X language in an unambiguous manner and thereby meet two important needs. First, those who develop and use IDEF1X models need a common understanding of the modeling constructs and rules. A precise definition of the meaning of the language components allows a model developed by one individual or group to be understood by another. Second, IDEF1X users must be supported in practice by automated tools that record and validate the models. Tool developers need a precise definition of the language so that their products assist users in applying the language correctly and allow exchange of models, at the semantic level, with other tools.

The purpose of IDEF1X as a modeling technique is the same as that of all modeling techniques employed in system analysis and development efforts, that is, to plan, build, or use systems and information systems in particular, it helps to understand the meaning of the concepts involved. Modeling provides a “language” for meanings and is sometimes referred to as closing the semantic gap between the concepts of the enterprise and the capabilities of the computer systems. Figure 1 summarizes the fundamental purpose of a model: to enable accurate and useful communication among users, analysts, and developers as they all reason about the same thing.

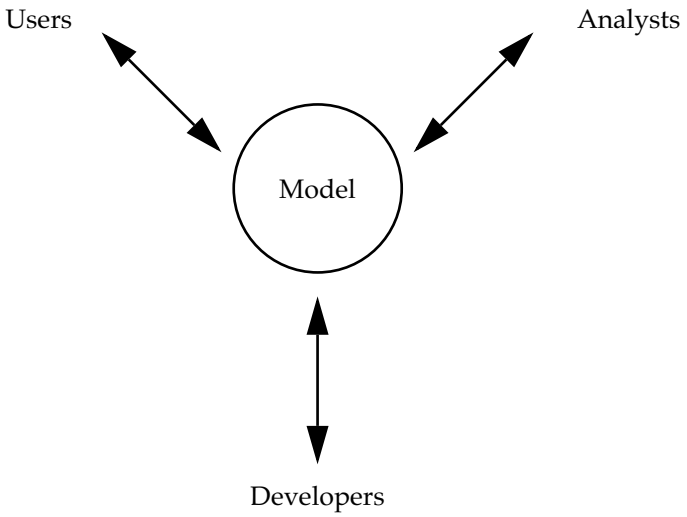

Figure 1—Communication of meanings

There are many uses for models, including process re-engineering, enterprise integration, detailed specification, implementation, and reverse engineering. Each is important.

1.3 Evolution of IDEF1X

The fundamental point of view originally adopted by IDEF1X was that the world was made up of interrelated things and that the meaning of data devolved from an understanding of these things and their relationships. This *key style* of IDEF1X modeling has been used over the past two decades to produce information models that represent the structure and semantics of data within an enterprise. The object model expands that point of view to include behavior. The evolution of IDEF1X has incorporated this goal of a broader understanding in the *identity-style* language introduced in this standard.

The transition from key-style to identity-style models involves bringing forward many earlier IDEF1X concepts, relaxing some of the restrictions, exploiting the fundamental concepts more fully, and adding important new concepts (see Figure 2). Each of the concepts used to produce an identity-style model is discussed fully in Clauses 4 through 8 of this standard. Clause 9 describes how to apply these concepts to produce a key-style model. Note that the concepts marked "unnecessary" in Figure 2 have been retained in the key-style language for backward-compatibility with existing models and for those who wish to continue producing key-style IDEF1X models.

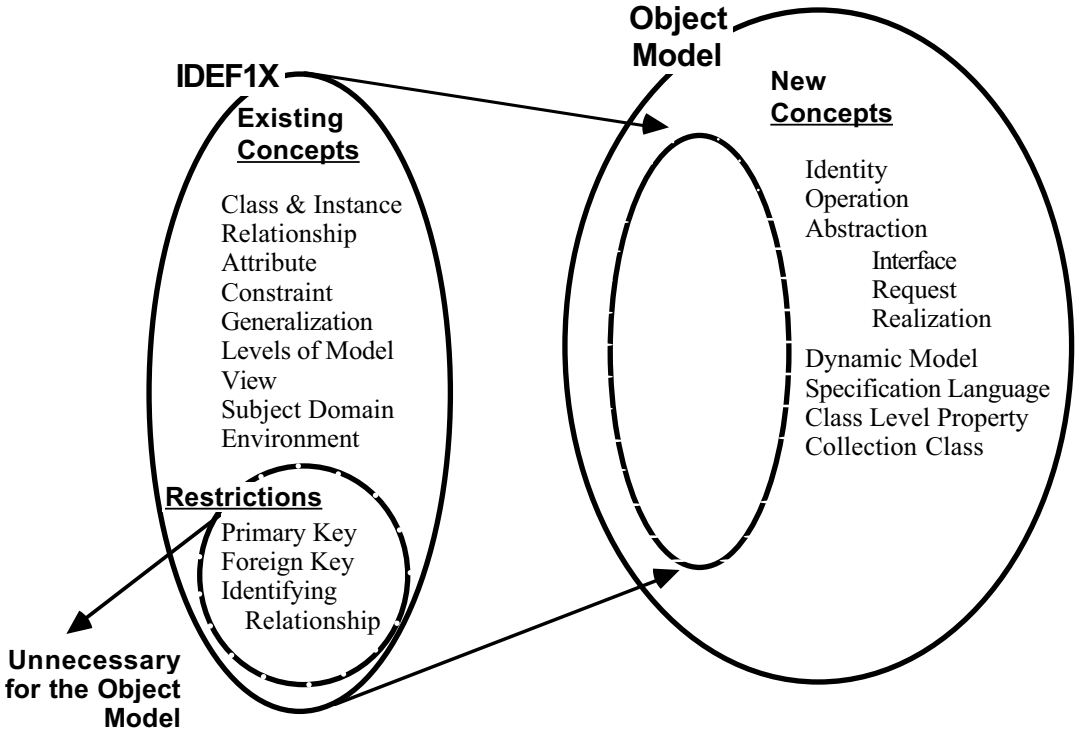

Figure 2—Correspondence of concepts

1.3.1 Understanding the data/process paradigm

The requirements for a modeling language are set largely by the way in which modelers choose to view the world. When IDEF1X was first developed in the early 1980s, the predominant system development view of

the world was framed in terms of data and processes. The modeling approach of this data/process (D/P) paradigm can be summarized as follows:

a) The world is made up of activities and things.
b) Things are integrated. Activities are free-standing.
c) Activities operate on things.

Within this approach, the primary objectives of an information modeling technique are

— To provide a means for understanding and analyzing an organization's data resources,
— To provide a common means of representing and reasoning about the data,
— To provide a method for presenting an overall view of the data required to run an enterprise,
— To provide a means for defining an application-independent view of data that can be validated by users and transformed into a physical database design,
— To provide a method for deriving an integrated data definition from existing data resources.

The D/P paradigm exerted a powerful and pervasive influence over all aspects of information technology. For IDEF the result was two distinct techniques: IDEF0 for process and IDEF1X for data. Thousands of successful systems have been developed using the D/P view of the world, and many developers continue to successfully employ the techniques.

1.3.2 Understanding the emerging object-oriented paradigm

The emergence of an object-oriented (OO) view of the world has strongly influenced the evolution of IDEF1X as described in this standard. The object paradigm takes a fundamentally different view of the world. In this paradigm, the modeling approach can be summarized as

a) The world is made up of objects.
b) Objects have knowledge and behavior.
c) There are no free-standing activities. Activity is accomplished by a collaboration of objects.
d) Knowledge and behavior are different aspects of the same object, considered *together,* behind an abstraction of responsibility.

Within this approach, the primary objectives of a modeling technique are

— To provide a means for understanding and analyzing the objects that are of interest to the organization,
— To provide a common means of representing and reasoning about these objects,
— To provide a method for presenting an overall view of the objects required to run an enterprise,
— To provide a means for defining an application-independent view of objects that can be validated by users and transformed into a physical design.

1.3.3 Contrasting the paradigms

The approaches of the D/P and OO paradigms are different. Major differences are summarized below in Table 1. For IDEF1X, the concepts that emerge from the D/P and OO approaches are not entirely incompatible. Indeed, there is a high degree of correspondence in the concepts.

While an IDEF1X model has typically been called a "data model," the term has always been something of a misnomer; an IDEF1X model was never a model of "data" *per se*. The entities in an IDEF1X data model are not "data" entities. An IDEF1X entity represented a concept, or meaning, in the minds of the people of the enterprise. To emphasize their concern with meaning as opposed to representational issues, "data models" like IDEF1X models are often called "semantic data models" or "conceptual models."

Table 1—D/P and OO approaches

D/P paradigm assumptions	OO paradigm assumptions	Contrast
An entity instance is a person, place, or thing (etc.) about which the enterprise needs to keep data.	An object is a distinct thing whose knowledge (data) or behaviors (processes) are relevant.	An object combines data and process (knowledge and behavior) and hides them behind an abstraction of responsibility.
There is no free-standing data. All data is organized around the shared real-world entities of the enterprise. The data is accessed by processes and shared across applications.	There is no free-standing knowledge (data). All knowledge is organized around the shared real-world objects of the enterprise. Each object maintains its own knowledge. The knowledge is available to (modifiable by) other objects upon request, across applications.	In D/P, processes directly access and change the data of an entity. In OO, an object must be asked for its knowledge; that knowledge is not directly accessible. Only the object itself can change its knowledge. Whether the object's knowledge is by memory or derivation is known only to the object.
Processes are free-standing. Process is organized around function, accesses entities, and is unique to an application.	There are no free-standing behaviors (processes). All behavior is organized around the shared real-world objects of the enterprise. Behavior is the responsibility of the object and available to other objects upon request, across applications.	In OO, all processing is accomplished by the actions of objects. An object acts by exploiting the knowledge and behavior of itself and collaborating objects via requests. Exactly what requests are made is known only to the object.
Similar entity instances are classified into classes, and classes are related by aggregation and generalization.	Similar objects (instances) are classified into classes, and classes are related by aggregation and generalization.	Essentially the same idea, except that the object class includes behaviors.
Each entity instance in a class is distinguishable from all others by its data values.	Each object is distinct from all other objects—it has an intrinsic, immutable identity, independent of its knowledge, behaviors, or class.	The OO model can recognize as distinct what the D/P paradigm treats as indistinguishable.
There are constraints on data.	There are constraints on both knowledge and behavior.	More general kinds of constraints are needed by the object model.
Rules are incorporated by defining processes that support them.	Rules are incorporated by defining behaviors that support them.	The D/P and OO paradigms both could be improved here. It would be better if rules could be disentangled from behaviors.

An *object model* is similarly a model of meaning, but it is a richer model that is closer to the ideal of a conceptual model. An object model attempts to capture the meanings of the knowledge and behaviors of objects. Yet, even state-of-practice object models still fall short of the ideal. The objects modeled are more like clerks than executives—they do what they are told to do but are short on vision and initiative. Objects await instruction ("Chris, put the pencil down.") rather than possessing the ability to utilize their knowledge to exhibit unrequested behavior. Nevertheless, object models are, in many environments, proving to be a major advance over the combination of separate process models and data models.

1.3.4 Expanded understanding of requirements

IDEF1X continues to meet the same requirements today that it was originally chartered to meet. However, leveraging on the capabilities that the OO approach offers, the understanding of those requirements has expanded. The expanded requirements can be summarized in terms of the five points of the "IDEF1X approach":

a) *Support the development of conceptual schemas*.

The conceptual schema has been characterized as those aspects of the enterprise that are invariant across the information products of the business and implementations of the enterprise business rules (application systems and databases). Previously, this scope had been understood to include only the grammar of the data. Now the understanding of the scope of the conceptual schema can be seen to include *operations* as well as *rules*.

In addition, the scope of platforms supporting applications designed using IDEF1X has broadened. In many areas, relational database management system-based applications are slowly giving way to ones built in some form of OO environment. For IDEF1X to remain transformable into functioning systems, the OO concepts must be incorporated so that the IDEF1X language is semantically broad enough to meet the needs of its users. IDEF1X needs to provide object modeling constructs appropriate for enterprise integration—from initial survey through implementation.

b) *Be a coherent language*.

IDEF1X has a clean, coherent structure with distinct and consistent semantic concepts. Many of the IDEF1X constructs have a graphical manifestation because their semantics can be easily captured that way and the resulting diagram easily read. However, as the language has evolved, not all concepts have been forced into an iconic representation; some concepts simply cannot be easily expressed graphically in a model that remains comprehensible.

In IDEF1X, as in any language, it is important that those things that are said most often are said easily, while allowing capture of those statements that are difficult to express graphically. Some constructs are best captured in text because the semantics being represented are inherently complex. Regardless of manifestation, graphical or textual, the language as a whole remains coherent and consistent.

c) *Be teachable*.

IDEF1X data modeling has been taught and practiced for nearly two decades. The teachability of the language has always been an important consideration. IDEF1X has served well as an effective communication tool across interdisciplinary teams. This rich body of experience and familiarity will not be lost. Data models created using previous versions of IDEF1X standards will continue to be conformant under this new version in the key-style language. An upward migration path for existing IDEF1X models and skill sets is provided, and training on the newer identity-style language is expected to emerge from the marketplace.

d) *Be well-tested and proven*.

The original elements of IDEF1X were based on years of experience with predecessor techniques and have been thoroughly tested both in US government development projects and in private industry. The identity style of IDEF1X introduced in this standard has been used in a variety of industry projects. Many of the features included in this version reflect requests and suggestions from IDEF1X practitioners, while others reflect the best features of the emerging object modeling techniques.

e) *Be automatable*.

IDEF1X consists of modeling constructs that can be precisely defined. The constructs of the identity-style model provide the basis for tool support for representation and reasoning about OO conceptual models, including direct execution of the models. With the formalization of the IDEF1X language, automated reasoning about the knowledge and behavior modeled is a realistic expectation.

IEEE Std 1320.2-1998 addresses the evolutionary needs of users of earlier versions of the language. Evolution is a process of change. A new version of a “creation” emerges and becomes dominant or dies out based on its suitability to the surrounding environment. During the transition, both versions of the creation will coexist.

So, too, both versions of IDEF1X (D/P and OO) are supported by this standard. The key style of IDEF1X is fully backward-compatible with FIPS PUB 184 [B13]; the use of the identity-style features is optional. Users can migrate as needed to the expanded semantic scope characteristic of the identity-style language.

1.3.5 IDEF1X in transition

The version of IDEF1X presented in this standard is based on the object model, which is the result of the confluence of three major branches of computer science: programming, database, and artificial intelligence. As of the mid-1990s, there was no single, authoritative source for what constitutes the object model, but there was a broad consensus on the core semantic concepts. Additional semantic concepts remain in flux, and there is little consensus on syntax and methodology.

The version of IDEF1X described in FIPS PUB 184 [B13] continues to be supported by this standard and is referred to here as $IDEF1X_{93}$. Where necessary to distinguish it from this earlier version, the extended IDEF1X defined in this standard (including both identity style and key style) is referred to as $IDEF1X_{97}$.

The constructs of $IDEF1X_{97}$ were developed by

a) Framing them in terms of organizing concepts congruent with the way people think,
b) Formalizing those concepts by assigning to each a mathematical construct such that formal operations on the constructs parallel correct reasoning about the concepts,
c) Specifying a notation (diagrams or language) that actively supports representation, communication, and reasoning in terms of the concepts.

The similarities between $IDEF1X_{93}$ and $IDEF1X_{97}$ are fundamental. For both, the world consists of distinct, individual things that exist in classes[2] and are related to one another.

$IDEF1X_{97}$ was developed by relaxing some of the restrictions in $IDEF1X_{93}$, exploiting the fundamental concepts more fully, and adding some important new concepts. Each of the semantic concepts of $IDEF1X_{93}$ has a corresponding identity-style $IDEF1X_{97}$ concept, but some of the $IDEF1X_{93}$ restrictions are not needed in identity-style $IDEF1X_{97}$. These restrictions are not basically in conflict with identity-style $IDEF1X_{97}$—they could be stated if there were any reason to do so. The goals and concepts of $IDEF1X_{93}$ are subsumed by $IDEF1X_{97}$; the essential semantic constructs of $IDEF1X_{93}$ are part of $IDEF1X_{97}$. Identity-style $IDEF1X_{97}$ includes concepts that are not present in $IDEF1X_{93}$.

The identity-style $IDEF1X_{97}$ concepts are object model concepts. $IDEF1X_{97}$ includes constructs for the distinct but related components of object abstraction—interface, request, and realization. Some of the specific concepts of $IDEF1X_{97}$ that support abstraction are the principle of substitutability, declarative constraints, and declarative specifications of properties.

The identity-style $IDEF1X_{97}$ constructs model objects over varying scopes and levels of refinement. $IDEF1X_{97}$ uses both graphics and a textual specification language. Its constructs are integrated with one another by a consistent, declarative approach to object semantics.

1.3.6 Future direction

The scope of this version of the IDEF1X language covers semantic data and object modeling. Use of this standard permits the construction of data and object models that may serve to support the management of concepts as a resource, the integration of information systems, and the building of computer databases and systems.

[2]Where some say “class” and “class instance” (or, “object”), this standard adopts the terminology “class” and “instance.”

1.3.6.1 Topics for future extensions

Aspects of the object model that are topics for future extensions of IDEF1X include the following:

a) *Dynamic models.* This version of IDEF1X covers the specification of both the interface and realization of active properties (operations) of a class. However, this version of this standard does not provide a set of graphics describing individual requests or patterns of requests.
b) *Transaction models.* There are many transaction models, and this version of IDEF1X has chosen not to select one but rather provide only the most basic notions of stating a constraint and providing a way to check it. Future versions of this standard may expand on the treatment of "transaction."
c) *Exception handling.* The specification of exception handling is an important aspect of many object languages. Future versions of this standard may incorporate exception handling into the language.

1.3.6.2 Features for expanded scope

In addition, the scope of the language may be expanded to include coverage for features frequently requested by IDEF1X users. Typical examples include

a) *Rules beyond constraints.* "Rule" is a more general, and more powerful, idea than constraint. This version of the standard deals only with constraints. A future version could incorporate a fuller treatment of "rules."
b) *Technology-dependent levels/default transformations.* An important characteristic of the original IDEF1X was the existence of a default transformation from a fully attributed model to an implementation in a database system such as IMS™, IDMS™, xBase, or relational.[3] In addition to database and object database transforms, the expanded coverage of IDEF1X suggests transforms into popular object languages such as Smalltalk™, C++, and Java™. From the overall management and development point of view, providing transforms into technology-specific models encouraged building models that are actually used. It created a very useful "practical" counterbalance to "wishful modeling." Enterprise integration does not come about because of modeling *per se*—the models have to be used. The existence of default transformations encourages use.

1.3.6.3 Constructs for future versions

Specific constructs to be incorporated into future versions include the following:

a) *Importing concepts.* Allow importing a concept defined in one environment into another environment.
b) *Importing types.* Allow importing a type defined in one view into another view.
c) *Initial values.* Allow the specification of initial values for instance-level and class-level attributes.
d) *Interfaces.* A *class* consists of an interface, which is a set of responsibilities, and a realization for each of those responsibilities. An *interface* consists of just a set of public responsibilities and, if specified independently, can be realized by many classes. A *type* is either an interface or a class. Add support for interfaces and types as distinct from classes.
e) *Ordered relationships.* Support the specification of the ordering of instances participating in relationships.
f) *References.* Support one-way mappings to state classes in a way that is symmetric with attribute and relationship mappings.
g) *Visibility.* Support the specification of the visibility of types and their responsibilities outside their defining view.

[3] All trade or product names are either trademarks or registered trademarks of their respective companies and are the property of their respective holders. The mention of a product in this document is for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products.

1.4 Conformance

This document is structured to permit its use in checking a model or modeling tool for conformance to this standard.

1.4.1 Identity-style model conformance

An identity-style model is conforming when

a) The lexical rules conform to Clause 4,
b) The class (state and value), generalization, and relationship semantics, syntax and rules conform to Clause 5,
c) The class (state and value), generalization, and relationship semantics conform to the semantics defined in Clause 10,
d) The responsibility semantics, syntax, rules, requests, and realizations conform to Clause 6,
e) The responsibility and realization semantics conform to the semantics defined in Clause 10,
f) The RCL conforms to the language syntax in Clause 7,
g) The RCL semantics conform to the semantics defined in Clause 10,
h) The model infrastructure constructs conform to Clause 8, and
i) The model instantiates the language metamodel in Clause 10.

1.4.2 Identity-style modeling tool conformance

An identity-style modeling tool is conforming when

a) The lexical rules conform to Clause 4,
b) The class (state and value), generalization, and relationship semantics, syntax and rules conform to Clause 5,
c) The class (state and value), generalization, and relationship semantics conform to the semantics defined in Clause 10,
d) The responsibility semantics, syntax, rules, requests, and realizations conform to Clause 6,
e) The responsibility and realization semantics conform to the semantics defined in Clause 10,
f) The RCL semantics conform to the semantics defined in Clause 10,
g) The model infrastructure constructs conform to Clause 8,
h) It can be demonstrated that the tool's metamodel maps to the language metamodel in Clause 10, that is,
    1) There is an *onto* mapping `lang` from the set of valid populations of the tool's metamodel to the set of valid populations of the language metamodel in Clause 10,
    2) There is a total mapping `tool` from the set of valid populations of the language metamodel in Clause 10 to the set of valid populations of the tool's metamodel, and
    3) For every valid population `L` of the language metamodel in Clause 10, `L = lang(tool(L))`,
i) It can be demonstrated that the tool correctly interprets RCL as specified in Clauses 7 and 10, and
j) Any tool extensions to the graphics or RCL can be demonstrated to be reducible to the graphics or RCL specified in this standard.

1.4.3 Key-style model conformance

A key-style model is conforming when

a) The lexical rules conform to Clause 4, and
b) The model components, semantics, syntax, and rules conform to Clause 8.

1.4.4 Key-style modeling tool conformance

A key-style modeling tool is conforming when

a) The lexical rules conform to Clause 4, and
b) The model components, semantics, syntax, and rules conform to Clause 8.