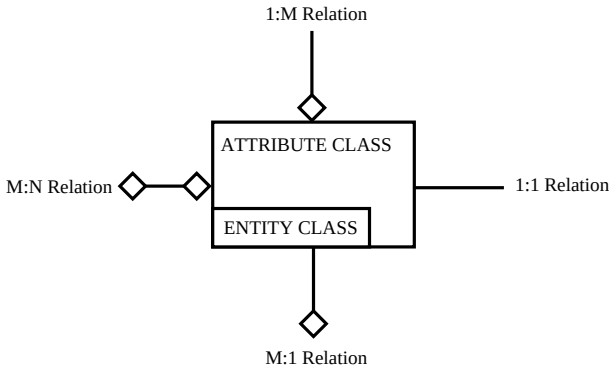
Section 6.0 Author's Guide to Creating IDEF1 Diagrams



6.0 Author's Guide To Creating IDEF1 Diagrams

6.1 Phase Zero - Context Definition

The IDEF1 information model must somehow be described and defined in terms of both its limitations and its ambitions. The modeler is one of the primary influences in the development of the scope of the model. Together, the modeler and the project manager unfold the plan for reaching the objectives of Phase Zero. These objectives include:

1. Project definition – a general statement of "what" has to be done, "why" and "how" it will get done.

2. Source material – a plan for the physical compilation of source material, including the indexing and filing of it.

3. Author conventions – a fundamental declaration of the conventions (optional methods) by which the author chooses to make and manage the model.

The products of these objectives, coupled with other descriptive and explanatory documents and information, become the products of the Phase Zero effort–the Phase Zero Kits. The purpose of these kits will be described more fully later in this section.

6.2 Project Definition

6.2.1 The Strategic Objective

The first activity in the modeling project is to establish where the project is going, how, and, to some extent, why. In this process, the project manager will establish his authority and mandate for the project. Some rough guidelines establishing the scope of the project in terms of money, time, and breadth of effort will have been laid down, usually in a project manager's authorizing documents. The extent of the project manager's responsibility, authority, and direct control must be made clear at this point in project start-up. The project manager must understand and establish this control, since it is the project manager's

responsibility to direct the activities of various other personnel during the course of the project.

On the basis of this established authority, the project manager selects the modeler and modeling team. Together they then formalize the scope of the project to be undertaken. A problem domain is first identified and then carefully defined.

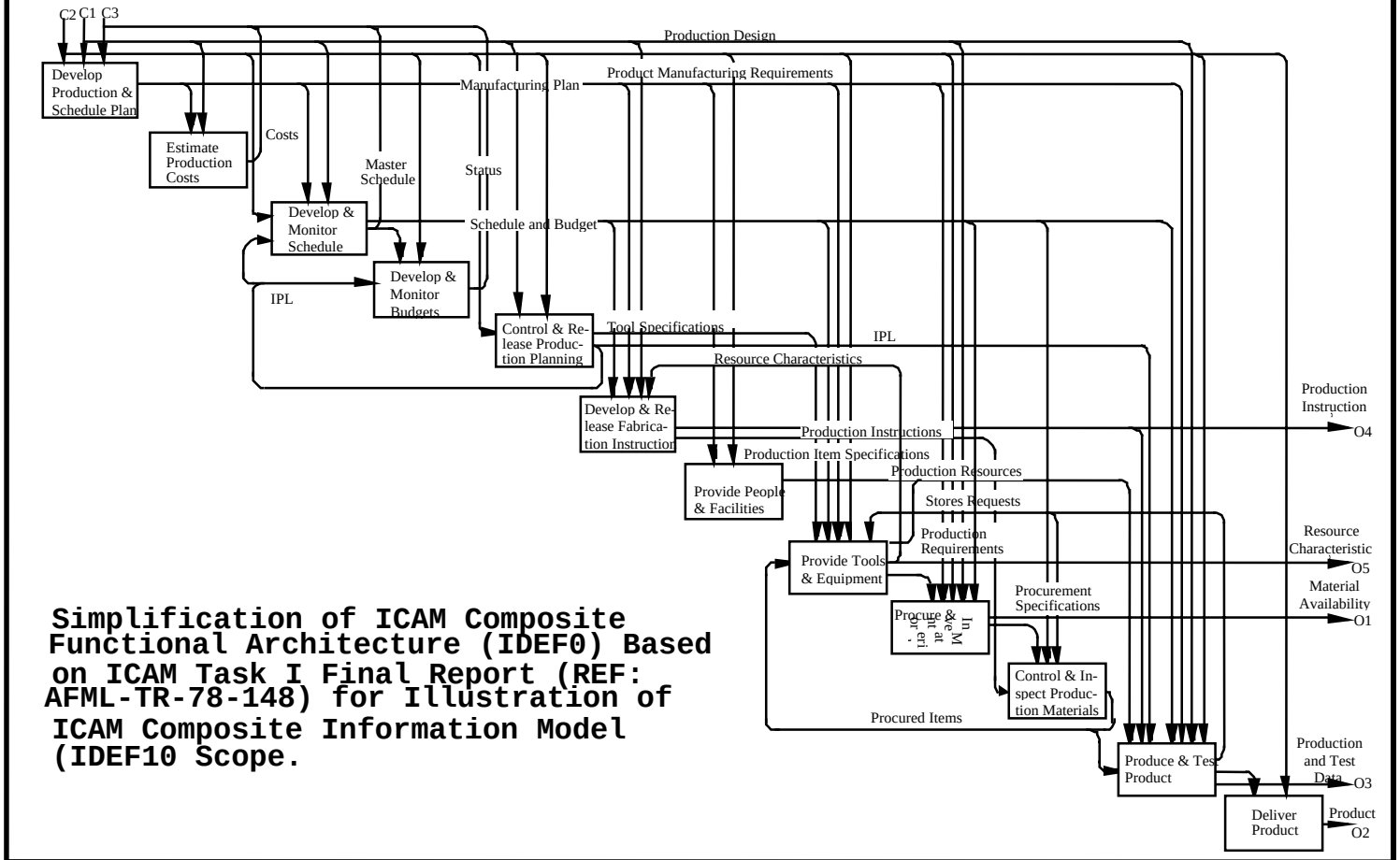
Thus, the number one element of the project plan emerges–the strategic objective. The strategic objective is comprised of two statements, one of intention and one of elimination. They are:

1. Statement of purpose – a statement that defines "what" the model will be concerned with, i.e., its contextual limits.

2. Statement of viewpoint – a statement which expresses the perspective of the model, the "eyes" through which the model is to be viewed.

One of the primary concerns which will be answered as a result of the establishment of the strategic objective is the concern over the time-frame reference for the model. Will it be a model of the current activities? Will it be a model of what is intended after a few changes in the enterprise? The strategic objective identifies the appropriate "time slice." Formal description of a problem domain for an IDEF1 modeling project may include the review, construction, modification, and/or elaboration of IDEFØ models. For this reason, both the modeler and the project manager must be versed, to some degree, in the authoring and use of IDEFØ models. Typically, an IDEFØ model exists which can serve as a basis for the problem domain. A sample of the scope of in IDEF1 model, as viewed from the IDEFØ perspective, can be seen in Figure 6-1.

USED AT:	AUTHOR: D.S. Coleman PROJECT: IMCV NODES: 1 2 3 4 5 6 7 8 9 10	DATE: 9 April 90 REV:	X WORKING ~~DRAFT~~ ~~RECOMMENDED PUBLICATION~~	READER: DATE:	CONTEXT



NODE: P0/F3	TITLE: IDEF1 Scope as Viewed from IDEF0 Perspective	NUMBER: DSC444

Figure 6-1. IDEF1 Scope

USED AT:	AUTHOR: I.M. Modeler DATE: 30 OCT 90 PROJECT: IDEF1 Work Station NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

I. Strategic Objective

Purpose

The purpose of this task is the development of an IDEF1 information model of the Purchase Requisition (Form PI-R6 4-72), as used to order parts and material which will be included in end products.

Viewpoint

The model will be developed from the viewpoint of a buyer in the Purchasing Department.

NODE: P0/T1	TITLE: Strategic Objective	NUMBER: IMM1

Figure 6-2. Strategic Objective

Differing viewpoints on a given problem invariably result in different models and it is crudely that the project manager and modeler reach an agreement on what purpose is being served, especially on what viewpoint is being used. As stated previously, the strategic objective is a juxtaposition of two statements describing the objectives: the purpose and the viewpoint. A sample strategic objective statement is shown in Figure 6-2.

6.2.2 Strategic Plan

The strategic plan is both a framing and organizing statement. It outlines the tasks to be accomplished and the sequence in which they should be accomplished. These are laid out in conformance with the overall tasks of the modeling effort:

1. Project planning
2. Data collection
3. Entity class definition
4. Relation class definition
5. Key Class definition
6. Attribute class population
7. Model validation
8. Acceptance review

Once the strategic planning effort gets underway, one of the several representation alternatives should be selected. These alternatives represent different ways in which the work efforts can be expressed in the strategic plan. Usually, modelers find it more convenient to combine methods rather than selecting only one by which to organize the project. Some of the more popular techniques employed include:

1. IDEFØ function models
2. WBS (work breakdown structures) – hierarchical or tree structure
3. Precedence networks (PERT)
4. Textual task definitions

A textual accompaniment to the formats which display the work structures is normally in order. It also takes a variety of forms, including text in paragraph style, text in outline form, text in tabular form, etc. A sample textual format is shown in Figure 6-3.

6.2.3 Functional Organization

In general, the model development may be viewed as ruled by the doctrine of "informed consent." By this we mean that the value of a model is measured not against some absolute norm but rather in terms of its acceptability to experts and laymen within the community for which it is built. This is accomplished through two mechanisms. A constant review by experts of the evolving model provides a measure of validity of that model within the particular environment of those experts. Then, a periodic review of the model by a committee of experts and laymen provides for a corporate consensus to the model. The principle of acceptability rather than perfection is maintained and information models are achieved which represent the enterprise in an acceptable and integrated fashion.

Another doctrine worthy of mention is the doctrine of "active intent." This means that, to the extent possible, the builders of a model are held responsible for what the model says. Nothing is assumed to have been left to the model reader's imagination. Nor is the reader at liberty to draw conclusions outside the scope of the statement of the model. This forces modeler to very carefully consider each piece of information added to the model and to carefully choose each figure which will be used to represent the model. A weakness in most techniques which this doctrine specifically attacks is the tendency to under-explain, or under-describe, components of the model. It is the intent of this doctrine that no imagination be required in the interpretation of in IDEF1 model.

USED AT:	AUTHOR:	DATE:	WORKING DRAFT	READER:	CONTEXT
	PROJECT:		RECOMMENDED PUBLICATION		
	NOTES: 1 2 3 4 5 6 7 8 9 10	REV:		DATE:	

Strategic Plan

Essentially, the strategy is based on two primary assertions:

A. The ability to develop a composite model which reflects real world characteristics is dependent on the participation of appropriate members of the aerospace manufacturing community.

B. The initial focus of the composite model should be on information used directly in the manufacturing effort.

These assertions, coupled with recommendations from individuals with experience in building various prototype information models, are reflected in the following statements:

PHASE 0

1. Evaluate IDEFØ, Function Model, and select 'target' functions representing the departure points for the composite modelling effort.

2. Identify the known entity classes in the information structure utilized or produced by the 'target' functions.

3. Develop a resource plan reflecting the manpower requirements, schedule, kit production rate, etc., and defining the degree of support needed from the aerospace manufacturing community.

NODE:	TITLE: Strategic Plan	NUMBER:

Figure 6-3. Strategic Plan

The functional organization is constructed to support these basic principles and to provide organized project controls.

The IDEF functional organization has five primary roles in it:

1. The project manager
2. The modeler
3. The sources of information
4. The expert reviewers
5. The acceptance review committee

Whole companies, departments, coalitions, teams, and others can serve in any one or several of the roles. The purpose of a role assignment, though, irrespective of the assignee, is the determination of responsibility. Each of these is further defined on the pages which follow.

One person may serve in more than one capacity in the functional organization of the modeling effort. But it is wise to remember that if there are insufficient points of view taken into account when finalizing the model, the model may represent a very narrow perspective. It may end up only partially serving to reach the objectives of the information modeling project.

In the cases of the project manager and the modeler, there must be a lead, or principal, individual who fulfills the role. Further, while it is the modeler's ultimate goal to have the model approved by the review committee, the modeler reports to the project manager, not the review committee. In this way the otherwise conflicting interests of the modeler, review committee, and project manager are disentangled. The project manager is always placed in a position of control, while the various technical discussions and approvals are automatically delegated to the qualified agency under that control. Figure 6-4 illustrates the functional project organization, with the project manager at the nucleus of all project activity.

Project Manager. The project manager is that person or organization identified as having administrative control over the modeling project. The project manager performs four essential functions in the modeling effort.

First of all the project manager selects the models. As a major part of this function, the project manager and the modeler must reach an agreement on the ground rules to be

followed in the modeling effort. These will include the use of this methodology, the extent of control the project manager expects to exercise over the modeler and the scope and orientation of the model to be developed.

The second function performed by the project manager is to identify the sources of information on which the modeler will draw to build the model. These sources may either be personnel particularly knowledgeable in some aspect of the manufacturing process, or documents which record, instigate, or

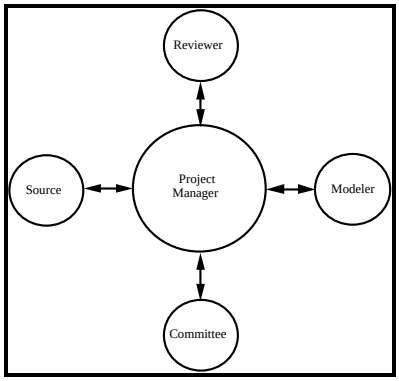


Figure 6-4. Functional Organization

report aspects of that process. From a modeling standpoint, personnel who can interpret and explain the information they deal with are the more desirable, but documents which record that information are usually less expensive to obtain. The project manager must be in a position to provide these sources to the modelers. Sources will initially be identified in modeling Phase Zero, but the list must be reviewed and revised as the effort progresses since the information required will tend to change as the model grows.

Next, the project manager selects experts on whose knowledge and understanding the modeler will draw for validation of the evolving model. Validation, as will be discussed under Experts, means concurrence that the model acceptably (in the abstract we would say "accurately" or "truly") reflects the system being modeled. The experts will be given portions of the model and asked to review and comment based on their particular knowledge. Clearly, more of an expert's time will be absorbed in the modeling effort than the time we would set aside for a source of basic information.

The initial list of experts will be established during Phase Zero, but will be reviewed and revised throughout the modeling effort as the need arises. Finally, the project manager forms and convenes the acceptance review committee. This committee periodically meets, under the chairmanship of the project manager, to consider issues of substance requiring arbitration, and to review portions of the model for formal acceptance. The project manager sits on the committee as its non-voting chairman, thereby providing the needed link between the modeler and the committee. While the modeler is not a member of the committee, it will frequently turn out that the project manager will invite the modeler to attend a committee meeting to provide background information or to explain difficult technical points. The first meeting of the committee is held during Phase Zero and thereafter at the discretion of the project manager.

Modeler. The modeler records the model on the basis of source material he is able to gather. It is the modeler's function to apply the modeling technique to the problem posed by the project manager. There are four primary functions performed by the modeler. These are: source data collection; education and training; model recording; and model control. The modeler is the central clearinghouse for both modeling methodology information and information about the model itself.

Before the modeler's primary functions begin, the modeler and the project manager study and establish the scope of the modeling effort. Then, the modeler outlines a project plan: the tasks required to reach the stated objectives. The project manager provides the modeler with a list of information sources and a list of experts on whom the modeler may rely. The modeler must ensure that the necessary lines of communication are established with all participants.

Source data are collected by the modeler from the various sources identified by the project manager. The nature of these data will depend largely on the modeling phase being exercised. Both personnel and documents will serve as sources of information throughout the modeling effort. With personnel especially with documents, the modeler must be particularly aware that each piece of source information provided represents a particular view of the information in the enterprise. Each producer and each user of information has its own distinct view of that information. The modeler is striving to see, "through the eyes of the sources," the underlying meaning and structure of the information. Each source provides a perspective, a view of the information sought. By compositing these views, by comparing and contrasting the various perspectives, the modeler must develop an image of the underlying

reality. Each document may be seen as a microcosmic implementation of a system meeting the rules of the underlying information model. The modeler attempts to capture all of these rules and represent them in a way that can be read, understood, and agreed upon by experts and informed laymen.

The modeler's second function is to provide assistance with the modeling technique to those who may require it. This will fall generally into three categories: general orientation for review committee members, sources, and some experts; model readership skills for some sources, and experts; and modeling skills for some experts and modelers, as required.

The third function performed is the recording of the model. The modeler records the model by means of filled-in modeling forms and diagram sheets. The methodology defines the rules for using and filling in these sheets.

The modeler also controls the development of the model. Files of derived source information are maintained to provide appropriate backup for decisions made by the modeler and to allow a measure of participation for both project manager and modeler use. This record of participation provides the modeler with an indication of the degree to which the anticipated scope is being covered. By knowing who has provided information in what areas and the "quality" of those interactions, the modeler can estimate the degree to which current modeling efforts have been effective in meeting the original goals.

The modeler is also responsible for periodically organizing the content of the model into some number of "Reader Kits" for distribution to reviewers. A reader kit is a collection of some number of pages of the model, organized to facilitate its review and the collection of comments from the information experts.

Sources. Source information for an IDEF1 model comes from every quarter within the manufacturing enterprise. These sources are often personnel who have a particular knowledge of some local area in manufacturing of management and whose contact with the model may be limited to a few short minutes of interview time. Yet these sources form the heart of the modeling process. It is their contribution which is modeled and their perception which provides the modeler with the needed insight to construct a valid, useful model.

The project manager identifies sources of information which may be effective based on the modeler's statement of need. As the modeling effort progresses, needs change and the list of sources must be revised. While the modeler must be careful to account for the information

provided by each source, both the modeler and source should be aware that any particular contribution is necessarily biased. Each source perceives the world a little differently and it is the modeler's responsibility to sort out these varying views. This is especially true of source documents.

Documents record the state of a minute portion of the enterprise at some point in time, but the information on a document is arranged for the convenience of its users and seldom directly reflects the underlying information structure. Redundancy of information is the most common example of this, but the occurrence of serendipitous information on a document is also a source of frequent and frustrating confusion. Documents are valuable sources of information for the model, but they require a great deal of interpretation, understanding, and corroboration to be used effectively.

Personnel used as sources can often extend themselves beyond their direct use of information to tell the modeler how that information is derived, interpreted, or utilized. By being prepared to ask appropriate questions, the modeler can use this information to advantage in understanding how the perception of one source may relate to that of another source.

Experts. An expert is a person appointed by the project manager who has a particular knowledge of some aspect of the manufacturing area being modeled and whose expertise will allow valuable critical comments of the progressing model. The impact that appropriate experts can have on the modeling effort cannot be overstressed. Both the modeler and the project manager should seriously consider the selection of each expert.

Experts are called on to critically review portion of the evolving model. This is accomplished through the exercise of some number of validation cycles (IDEF Kit Cycle), and by the use of reader kits. These kits provide the expert with a related collection of information presented so as to "tell a story." In this fashion the expert is provided the information in an easily digestible form and is challenged to "fill in the blanks" or "complete the story." While the kit is largely based on modeler interpretation of information from informed sources, the comments of experts may also be expected to provide high quality source material for the refinement of the model. The particular expertise of these people makes them uniquely qualified to assist the modeler in constructing and refining the model. The modeler must take every opportunity to solicit such input and this is why the kits of information must present the expert with concise, clear problems to solve relative to the modeling effort.

The primary job of the expert is to validate the model. Expert validation is the principal means of achieving an informed consensus of experts. A valid model is one agreed to by experts informed about the model. Note that it is not necessary for a model to be "right" for it to be valid. If the majority of experts in the field agree that the model appropriately and completely represents the area of concern, then the model is considered to be valid. Dissenting opinions are always noted and, it is assumed by the discipline that models are invalid until proven otherwise. That is why expert participation is so vital to the modeling effort. When the modeler first constructs a portion of the model he is saying: "I have reviewed the facts and concluded the following..." When that portion is subsequently submitted to experts for review, he asks, "...Am I right?..." Expert comments are then taken into account in revising that portion of the model which the experts do not agree with, always bearing in mind that a consensus is being sought.

Experts, more than any other non-modeling participants, require training to be effective. In fact, one of the modeler's responsibilities is to ensure that experts have an adequate understanding of the modeling methodology and process. Principally, experts require good model readership skills, but periodically it will be helpful to train an expert in some of the rudiments of model authorship. By providing experts with a basic understanding of modeling the project is assured of useful input from those experts. Further, the stepwise, incremental nature of the modeling process presents experts with the modeling methodology in "small doses." This tends to enhance the expert's ability to understand and contribute to the modeling effort.

Acceptance Review Committee. The acceptance review committee is formed of experts and informed laymen in the area of manufacturing to be addressed by the modeling effort. The project manager forms the committee and sits as its chairman. It is the function of the review committee to provide guidance and arbitration in the modeling effort and to pass final judgement over the ultimate product of the effort: an IDEF1 information model. Since this model is one part in a complex series of events to determine and implement systematic improvements in the manufacturing productivity of the enterprise, it is important that the committee include ample representation from providers, processors, and end users of the information represented. Very often this will mean that policy planners and data processing experts will be included on the committee. These personnel are primarily concerned with eventual uses to which the model will be put. It may also be advantageous to include on the committee experts from manufacturing or management areas outside, but related to, the

area under study. These personnel often can contribute valuable insight into how the information model will impact, or should be impacted by, ongoing work in other areas.

It is not uncommon for personnel who serve in the role of experts to also serve as members of the review committee. No conflict of interest should be anticipated. An expert is often only exposed to restricted portions of the model at various intermediate stages, but the review committee must pass judgement on the entire model. It is much less common for personnel who serve in the role of source to also sit on the committee; their knowledge is usually so restricted in coverage that it excludes them from practical contribution to the committee. It is ill-advised for modelers to sit on the committee, because a severe conflict of interest is clearly evident. The role of the modeler is to record the model without bias, while the role of the committee is to ensure that the model in fact represents their particular manufacturing enterprise.

The end product of this segment of the project definition is the documentation of specific assignments made by the project manager to fulfill each of the functional role requirements of the modeling technique. Once again, there are a number of graphic and textual alternatives available to the modeler for representing these assignments.

6.2.4 Resource Allocation

The final effort in the Project Definition portion of Phase Zero is the development of the resource allocation plan. This plan stipulates the magnitude of the task effort, the time frame through which the task will be conducted, and the sequence of the efforts involved.

The major activities within the modeling effort must be broken down into more manageable levels of detail. These more manageable pieces are then scheduled and phased and then a milestone chart of these scheduled events can be produced.

The resource allocation plan displays the required hours of effort anticipated for each participant in the project in relation to each phase of model development and calendar time frame. It includes a summation of estimated hours by modeling phase and role as well as a summation of estimated hours for each phase and for the total project.

Any one of a number of alternative formats can be selected to express the resource allocation arrangement. Text, bar charts, milestone charts, etc., can fulfill the requirement to publish the details of the resource allocation plan.

6.2.5 Source Material – Data Collection Plan

One of the first problems confronting the modeler is the determination or what sort of material needs to be gathered and from what sources it should be gathered. Often the scope and context of the IDEF1 model will be determined based on an analysis of an IDEFØ function model. Once the analysis of the functions and pipelines between functions is completed, "target" functions within the enterprise represented by the function model can be identified. A "target" function node is one that represents a concentration of information in use which is representative of the problem domain. An example of "target" function node selection is shown in Figure 6-5.

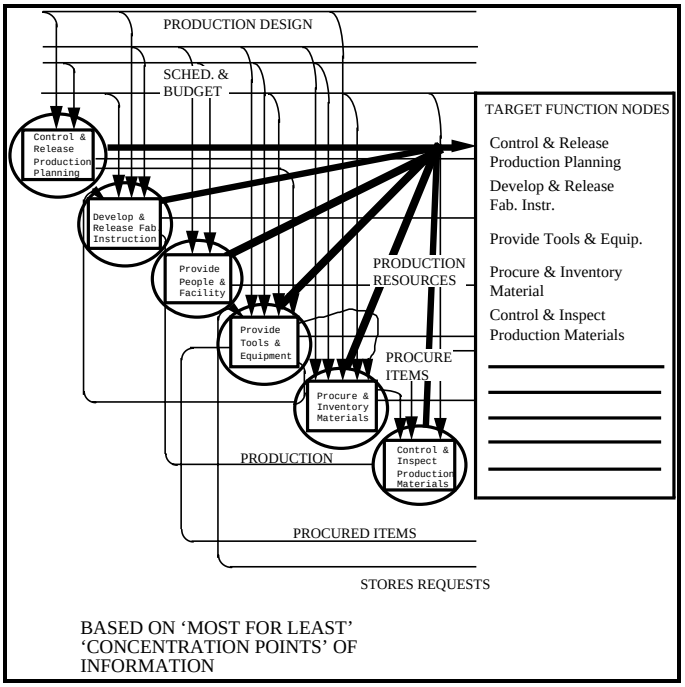


Figure 6-5. Target Function Nodes

Once the target functional areas have been identified and the primary information categories of interest selected individuals within functions can be selected to participate in the data gathering process. This data gathering can be accomplished in several ways, including interviews with knowledgeable individuals; observation of activities; evaluation of documents, policies, and procedures; application specific information models; etc. This requires translation of the target function nodes into their equivalent, or contributing,

administrative participants. Once the administrative groups participating in a "target" function have been identified, the project manager can proceed to identify individuals (or specific observable areas) which can be used as sources of material for the model. A representation of the translation of "target" function nodes into administrative and individual participants is shown in Figure 6-6.

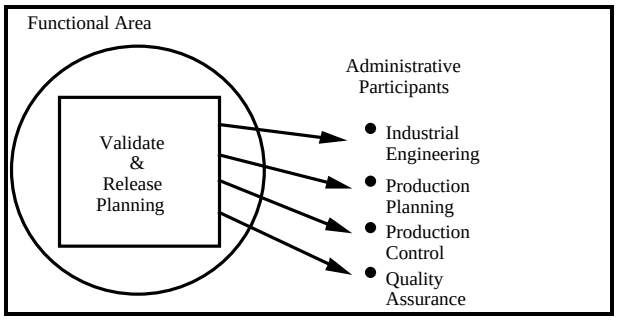


Figure 6-6. Identifying Participants

Regardless of the method used, the objective of the modeler at this point is to establish a plan for the collection or representative documentation reflecting the information pertinent to the purpose and viewpoint of the model. Once collected, each piece of this documentation should be marked in such a way that is traceable back to its source. This documentation, along with added documentation which is discovered through the course of the modeling, will be constantly referenced in the various phases of model development. It is this source material which the modeler will study and search for objective evidence lending credibility to the basic structural characteristics of the model and meaning of the information represented.

The source material may take on any one of several forms and may be fairly widespread throughout an organization. It may take the form of:

1. Document (various media)
2. Blank forms (supplies)
3. Policies and Procedures (instructions)
4. Interview results
5. Observation results, etc.

To accomplish the objective successfully, a sound data collection plan is of paramount importance. This data collection plan must reflect what kind of information is of importance, where that information is available, and who will supply it. Characteristic of Phase Zero, this plan may take any of several forms of presentation, as long as the basic information requirements are met.

As the source material is actually collected, its availability is recorded on the source material log. The source material log is the primary index to all source material used in the project. Each piece of source material is sequentially assigned a unique identifying number as the log is filled out. This number is written on the source material prior to its being placed in file. A sample of the source material log is shown in Figure 6-7.

Once the collection of source material is initiated, the source data identification can commence. This is a list of every name (i.e., every element) that is referenced in the source material. Each name occurs only once on the source data list and each name must be traceable to the specific piece(s) of source material where it can be observed. A sample source data list is shown in Figure 6-8. An example of how this traceability is employed is shown in Figure 6-9.

There are two work activities represented here:

1. Preparation of the data collection plan
2. Execution of the data collection plan

USED AT:	AUTHOR: I.M. Modeler DATE: 30 OCT 90 PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

Material #	Source Material Name/Description	Received From	Comments
SM#1	Purchase Requisition/Form PI-R6 4-72	U.R. Buyer	
SM#2	Procedure #079-003 /Rev. 00 " Preparation of the Requisition"	U.R. Buyer	
SM#3	Procedure #079-001/ Rev. 00 " Preparation of the Purchase Order"	Policy and Procedures Manual	
SM#4	Procedure #101-506 " Purchasing Codes"	Policy and Procedures Manual	
SM#5	B.J. Commodity Code List	U.R. Buyer	
SM#6	B.J. Product Code List	U.R. Buyer	
SM			
SM			
SM			
SM			
SM			

NODE: P /X1	TITLE: Source Material Log	NUMBER: IMM5

Figure 6-7. Source Material Log

USED AT:	AUTHOR: I.M. Modeler DATE: 30 OCT 90 PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	~~WORKING DRAFT~~ ~~RECOMMENDED PUBLICATION~~	READER: DATE:	CONTEXT

Source Data #	Source Data Name	Source Material Cross-Reference	Comments
SD#1	Requisition Number	SM#1,#27,#36	Pre-Printed on Form
SD#2	Buyer Code	SM#1,#17	
SD#3	Vendor Number	SM#1,#21,#27	
SD#4	Order Code	SM#1	Only for Orders Not Delivered to Plant 1 or 3 See 079-001
SD#5	Chg. No. (Change Number)	SM#1	
SD#6	Ship To (Location)	SM#1	
SD#7	Purchase Requisition	SM#1	
SD#8	Vendor Name and Address	SM#1	
SD#9	Non-Confirming/Confirming To	SM#1	Name of Person Contacted Entered in Space Provided
SD#10	Extra Copies	SM#1	For Extra Purchase Order Copies
SD#11	Requester (Name)	SM#1	

NODE: P /X2	TITLE: Source Data List	NUMBER: IMM6

Figure 6-8. Source Data List

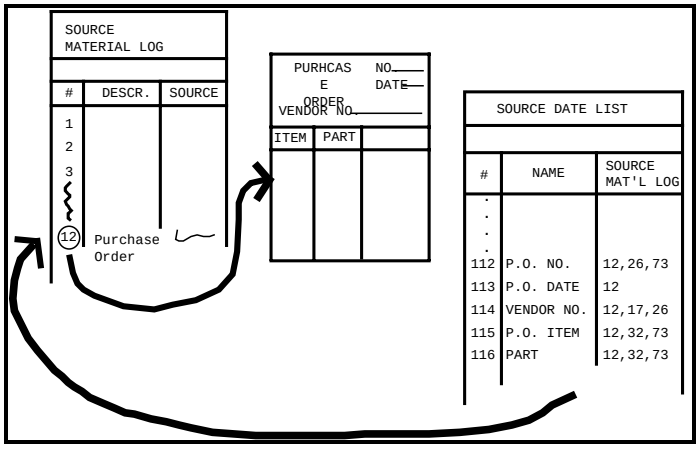


Figure 6-9. Traceability of Source Data

The modeler includes the actual data collection plan in the Phase Zero documentation which is distributed to readers, but the source material collected (forms, records, etc.) is not distributed for review. Only the Source Material Log and Source Data List are distributed as a part of the Phase Zero documentation.

6.2.6 Author Conventions

Author conventions are those latitudes granted to the modeler (author) to assist in the development of the model, its review kits, and other presentations. Their purpose is specifically for the enhancement of the presentation of the material. They may be used anywhere that would facilitate a better understanding and appreciation of any portion of the model.

Author conventions may take on various forms and appear in various places. For example, a textual representation of a strategic plan rather than a Work Breakdown Structure or other type of chart. But the most important aspect of all of this is what author conventions are not.

Author conventions are not:

1. Formal extensions of the technique
2. Violations of the technique

Author conventions are developed to serve specific needs. Each convention must be documented as it is developed and included in the Phase Zero documentation which is distributed for review.

6.2.7 Phase Zero Kits

Phase Zero kits are assembled in groups. The first group contains the products of the four primary stages of project definition–strategic objective, strategic plan, functional organization, and resource allocation. These are referred to as the project planning kits. Next are the source material kits. This group consists of the Data Collection plan, the Source Material Log, and the Source Data List. Finally, there is the last group of kits, the Author Conventions.

There will typically be some number of kits representing each group in Phase Zero. Together, the kits represent the whole of the Phase Zero product. Each of these kits is prefaced by a Kit Cover Sheet. A thorough description of the forms and procedures for submitting kits is found in Section 5.0.

The Phase Zero kit structure is not complicated. Basically, a Phase Zero kit is composed of a kit cover sheet, followed by some number of pages representing one or more sections of the Phase Zero documentation. An example of the Phase Zero kit structure is reflected in Figure 6-10. Most of the pages in Phase Zero documentation are textual in nature. Each page of the textual material is assigned a unique “node” number within the phase. Figure 6-11 shows an example of this standard.

It is imperative that Phase Zero content be approved by the project manager prior to initiating Phase One. Later on, revisions may be made to this phase to reflect the appearance of new elements in the model, but new Phase Zero kits have to be issued then to document the change.

6.3 Phase One – Entity Class Definition

The objective of Phase One is to identify and define the entity classes which fall within the problem domain being modeled. The first step in this process is the identification of entity classes.

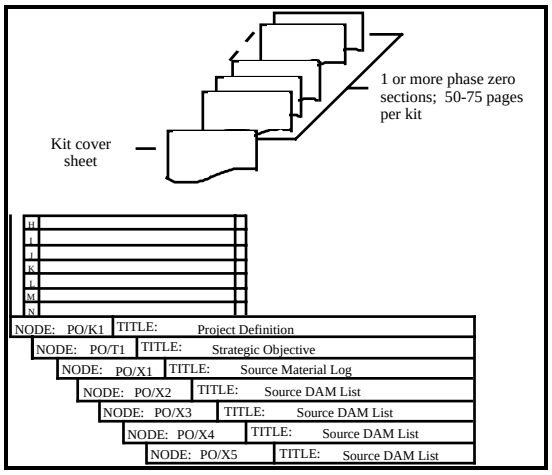


Figure 6-10. Phase Zero Kit Structure

6.3.1 Entity Class Definition

Most of these should have been represented on the Source Data List constructed during Phase Zero. The modeler must first identify within the list of names those things which represent potentially viable entity classes. One way this can be simplified is to identify the occurrences of all nouns in the list. For example, terms like “part,” “vehicle,” “machine,” “drawing,” etc., would be considered potentially viable as entity classes. Another method is to identify those terms ending with the use of the word “code” or “number;” for example, part number, purchase order number, routing number, etc. The phrase, or word, preceding the word “code” or “number;” could also be considered at this stage a potentially viable entity class. For the remainder of the items on this list, the modeler must ask whether the word represents an object about which information is known, or is information about an object. Those items which fall into the category of being objects about which information is known may also be viable entity classes.

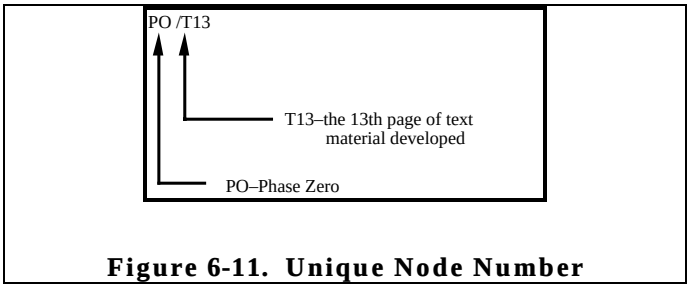


Figure 6-11. Unique Node Number

Entity classes result from a synthesis of basic entities, which become members of the entity class. This means that some number of entities, all of whose detailed pieces are the same, are represented as an entity class. An example of this concept is shown in Figure 6-12. The entity classes, although expressed in a two-dimensional form, must be thought of in terms of a three-dimensional image. The third dimension is the membership mentioned earlier. Each occurrence of an entity in an entity class is a member of the entity class, all with the same kind of identifying information.

At the end of this process of selection of terms, the modeler has completed what is referred to as the entity class pool. This pool simply contains all of the known names of entity classes within the context of the model being constructed at this point. As the modeler is building the entity class pool, a discrete identification number is assigned to each entry and reference to its source is recorded. This way, the traceability of the information is maintained. The integrity of the pool remains intact and the management of the pool is relatively easy. A sample of an entity class pool is shown in Figure 6-13.

It is unlikely that all names on the list will remain as entity classes by the end of Phase Four. Also, a number of new entity classes will be added to this list and become a part of the information model as the modeling progresses and the understanding of the information improves.

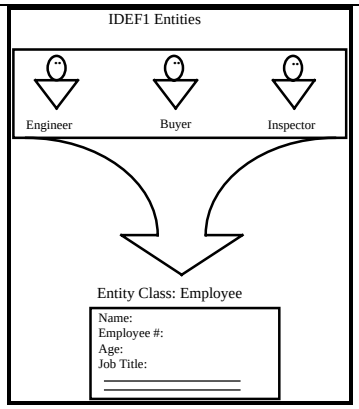


Figure 6-12. Synthesizing an Entity Class

USED AT:	AUTHOR: I.M. Modeler PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 30 OCT 90 REV: 1 NOV 90 3 NOV 90	WORKING DRAFT ~~RECOMMENDED~~ PUBLICATION	READER: DATE:	CONTEXT

Entity Class Node No.	Name	Source Data ID #	Entity Class Node No.	Name	Source Data ID #
E1	Purchase Requisition	SD1	E18	Bill of Material	SD38
E2	Buyer	SD2	E19	Route Sheet	SD40
E3	Vendor	SD3	E20	Destination	SD41
E4	Purchase Order	SD15	E21	Approver	SD43
E5	Ship To Location	SD6	E22	Part Source	SD
E6	Requester	SD11	E23	Confirmer	SD
E7	Department	SD12	E24	Extra Copy	SD
E8	Pattern	SD21	E		SD
E9	Part	SD26	E		SD
E10	Purchase Req. Item	SD23	E		SD
E11	Commodity	SD30	E		SD
E12	Purchase Req. Line	SD31	E		SD
E13	Job	SD34	E		SD
E14	Account	SD36	E		SD
E15	Product	SD37	E		SD
E16	B.M. Page	SD38	E		SD
E17	B.M. Line	SD39	E		SD

NODE: P1/X1	TITLE: Entity Class Pool	NUMBER: IMM11

Figure 6-13. Entity Class Pool

Entity class names discovered in some later phase must be added to the entity class pool and assigned a unique identification number. One of the products of the Phase One effort is the entity class pool. To remain viable it must be up to date.

The next product that will emerge out of the Phase One efforts is the beginning of the Entity Class Glossary. During Phase One, the glossary is merely an assemblage of the entity class definition pages.

There is a standard outline for the entity class definition page, which is reflected below. The heading and title blocks of the form are standard, but the body of the form is unique to the entity class definition page. Each of the items contained in the body of the page is discussed below:

1. Entity Class Name – Entity class name is the formal name by which the entity class will be recognized in the IDEF1 model. It should be fully descriptive in nature, not shorthand.

2. Entity Class Label – The entity class label is a pseudonym assigned to the entity class which is based on the entity class name. It should bear some resemblance to the entity class name, but is a short form of the name. It is destined to be the form of the name used in the entity class box throughout the model. It is stressed that, although the entity class label is a shorter form of the entity class name, it must be meaningful. Character strings which are unrecognizable or carry no immediate recognized significance to the reader are worthless as entity class labels.

3. Entity Class Definition – This is a definition of the entity class which is most commonly used in the enterprise from the viewpoint upon which the model is based. It is not intended to be "Websterian" (dictionary-like) in nature. It would be meaningless (if not totally confusing) to include definitions outside of the Phase Zero scope, since the meaning of the information reflected in the model is specific to the viewpoint upon which the model is based and to the context of the model defined in Phase Zero There may be slight connotative differences in the way that the entity class is defined, primarily based on contextual usage. When these occur, or when there are alternate definitions (which are not necessarily the most common from the viewpoint of the model), these should also be recorded. It is up to the reviewers to identify what definition should be associated with the term used to identify the entity class. The Phase One definition process is the mechanism used to force the evolvement of a commonly accepted definition.

4. Entity Class Synonyms – This is a list of other names by which the entity class might be known. The only rule pertaining to this is that the definition associated with the entity class name must apply exactly and precisely to each of the synonyms in the synonym list.

A sample of the entity class definition page is exhibited in Figure 6-14.

Entity class definitions are not most easily organized and completed by first going after the ones which require the least amount of research. Thus, the volume of glossary pages will surge in the shortest period of time, then the modeler can conduct the research required to become fully conversant with the rest of the names in the pool. Good management of the time and the effort required to gather and define the information will ensure that the modeling effort continues at a reasonable pace.

Each element of the entity class glossary page is defined as it pertains to the methodology and its use in this instance. Their definitions and an example of a "good" and a "poor" application of each follows:

1. Number – sequential; non-significant; assigned in the entity class pool.

2. Name – descriptive; a singular noun; must not be an abbreviation

   Good: Purchase Order

   Poor: Purch. Ord.

3. Label – meaningful, but in short-form; tied closely with name to ensure compatibility and recognition.

   Good: Purch. Ord. or Prod. Order

   Poor: P.O.

4. Definition – precise; specific; complete; "universally" understandable.

   Good: Employee: a person who performs specific duties for the company over a duration of 20 or more hours per week and meets the other stipulations of an employee as defined by the state employment office.

   Poor: Employee: a person who works for the Company.

5. Synonyms – other names for the entity; must be 100% congruous in meaning; must be synonymous in context.

   Entity Class Synonyms

   Good: Purchase Order Purchase Authorization

   Purchase Agreement

   Poor: Purchase Order Purchase Request

Definition of the entity classes can commence after the construction of the entity class pool. Each entity class will have one definition page which contains the information about the entity class that is known at the time.

A completed entity class definition page is exhibited in Figure 6-15.

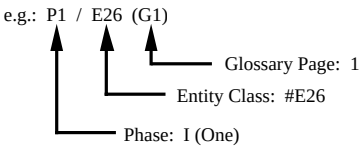
6.3.2 Phase One Kits

The modeler is now ready to construct and circulate Phase One kits to the export reviewers. Eventually, all members of the entity class pool will be defined and circulated through Phase One to the expert reviewers, but the modeler is most interested in establishing adequate definitions to circulate in kit form as early as possible. The modeler will therefore select some number of these potential entity classes, usually 15 to 20 per kit, define them, and release the kit for review and comment. While the review and comment cycle is going on, the modeler will be defining additional entity classes and preparing additional kits which will themselves be circulated for additional review and comment as well. The number of kits to be circulated during this phase is determined basically by two things:

1. The total number of entity classes.
2. The amount of revision and iteration required on the entity class definitions.

The standard diagram form, previously described, is used as the basis for the Phase One kits. The entity class definition page identification follows a prescribed pattern. Included are:

1. Node – a number comprised of the phase number, entity class number, and glossary page number...



2. Title – "Entity Class Definition:" followed by the entity class name...

   e.g.: Entity Class Definition: Purchase Order

3. Number – Author Page Control (C) Number

An exmple of the entity class definition page labeling convention is shown in Figure 6-16.

USED AT:	AUTHOR:	DATE:	WORKING DRAFT	READER:	CONTEXT
	PROJECT:		RECOMMENDED PUBLICATION		
	NOTES: 1 2 3 4 5 6 7 8 9 10	REV:		DATE:	

Entity Class Name:

Entity Class Label:

Entity Class Definition:

Entity Class Synonym(s):

NODE: P1/E1 (G1)	TITLE: Entity Class Definition:	NUMBER:

Figure 6-14. Entity Class Definition Page

USED AT:	AUTHOR: I.M. Modeler DATE: 30 OCT 90 PROJECT: IDEF1 Work Shop NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

Entity Class Name: Purchase Requisition

Entity Class Label: Purch.Req.

Entity Class Definition: A Purchase Requisition reflects information which is used by the Inventory Control Department to request the Purchasing Department to

Entity Class Synonyms:

NODE: P1/E1 (G1)	TITLE: Entity Class Definition: Purchase Requisition	NUMBER: IMM12

Figure 6-15. Completed Entity Class Definition Page

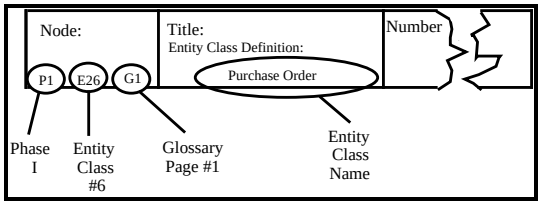


Figure 6-16. Entity Class Definition Page Labeling

As indicated earlier, a Phase One kit typically consists of 15-20 entity class definition pages. Each kit is prefaced by a standard kit cover sheet. A sample of a Phase One kit cover sheet, as completed by the modeler, is exhibited in Figure 6-17.

The structure of Phase One kits is not complicated. Each Phase One kit is composed of 15-20 entity class definition pages, prefaced by a kit cover sheet. This structure is exhibited in Figure 6-18.

Structuring the Phase One kits of about 15 to 20 entity classes per kit will provide, on the average, 1 to 1 1/2 hours of review work for each expert per kit.

Once structured, the modeler must ensure that adequate copies of the kit can be reproduced to meet the distribution requirements of the validation cycle. Then the originals of each page are placed in the "current model file" until required again.

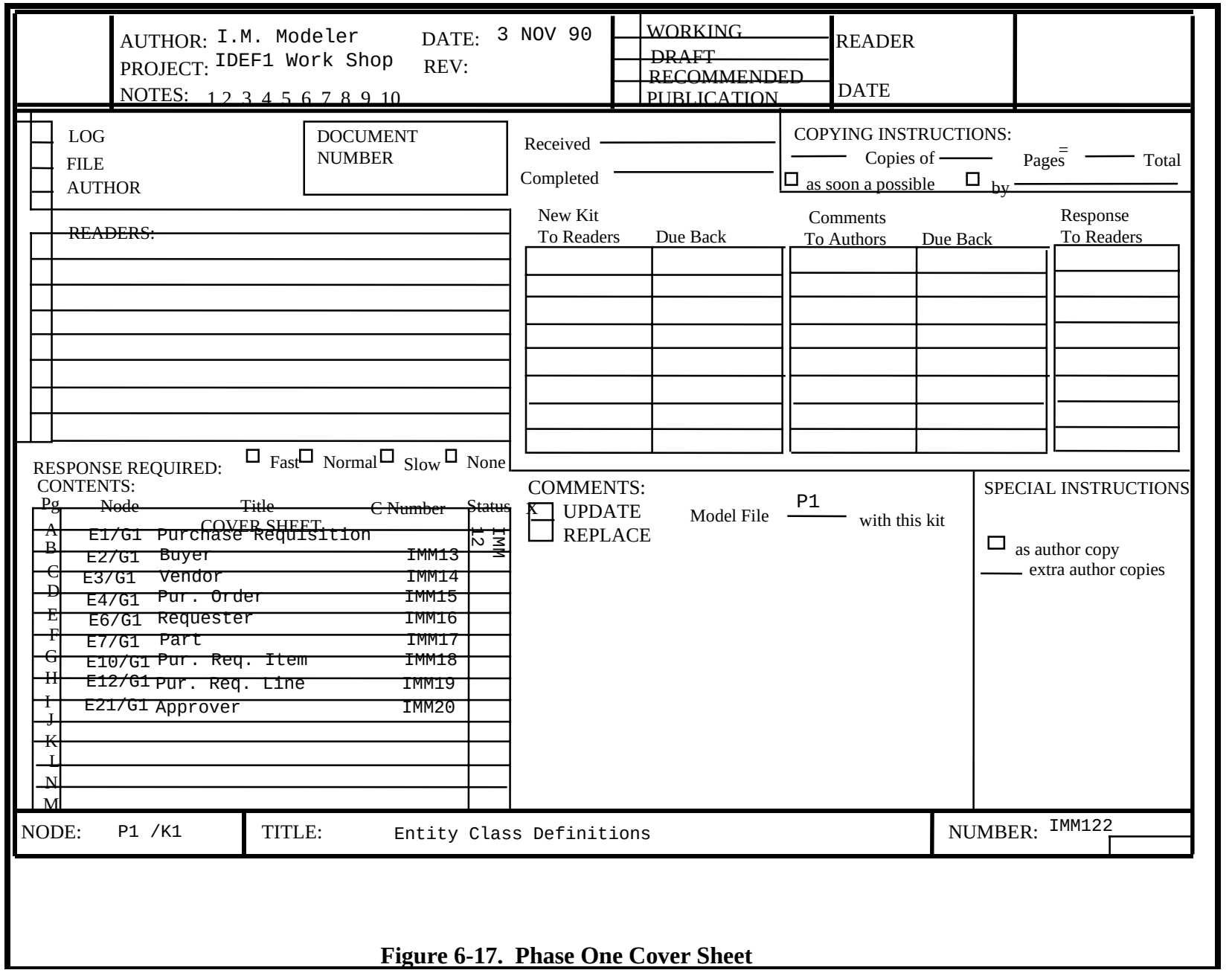


Figure 6-17. Phase One Cover Sheet

6.4 Phase Two – Relation Class Definition

The objectives of this phase in the development of the model are threefold, two of which have to do with the identification and definition of relationships and their representation as relation classes. With the relation call definitions in hand the modeler can proceed with the third objective: the construction of the entity class diagrams. All of the products of Phase Two will emerge as the by-products of the effort to reach these objectives. They will finally be joined together in the Phase Two kits, which serve as the vehicle for validation of the model.

Following is a list of the products which result from the Phase Two effort.

1. Relation Matrix
2. Entity Class Diagrams
3. Relation Class Definitions
4. Entity Class Node Cross-reference
5. Reference Diagrams (FEOs)
6. Phase Two Kits

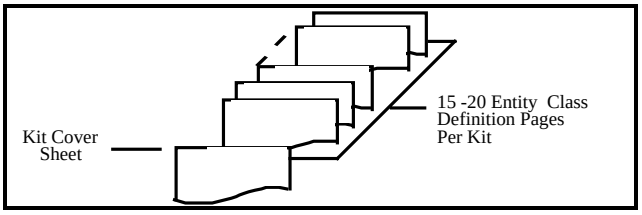


Figure 6-18. Phase One Kit Structure

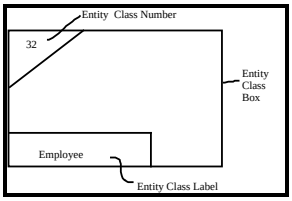


Figure 6-19. Phase Two Entity Class Box

To get started on this effort requires that the modeler become intimately familiar with the basic structural figure of modeling–the entity class box. A sample is shown in Figure 6-19. The symbol of an entity class is a simple rectangle with a small rectangle in the lower left hand corner for the entity class label, and a small triangle in the upper left hand corner for the entity class number.

The basic format used in the model diagrams is two entity class boxes connected by a straight line. The line represents relationships between entities. The basic diagram format is illustrated in Figure 6-20. This relationship is defined by adding symbols and phrases to the connecting line. This process results in the identification of what is called the Relation Class.

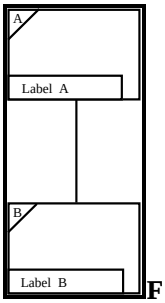


Figure 6-20. Entity Class Diagram Syntax

The initial step in Phase Two requires that the modeler understand the basic purpose served by the Relation Class.

First of all, a relationship is a meaningful association between two entities. This is illustrated in Figure 6-21. Here the relationship is between "Operator #862" and "Machine #12678." There are literally millions of these kinds of relationships pictured everyday–between a car and a driver, an aircraft and a pilot, a system and a procedure, and so on.

A relation class is the manner in which members of one entity class are associated with (related to) members of another entity class (or to members of the same entity class). This is illustrated in Figure 6-21, as well.

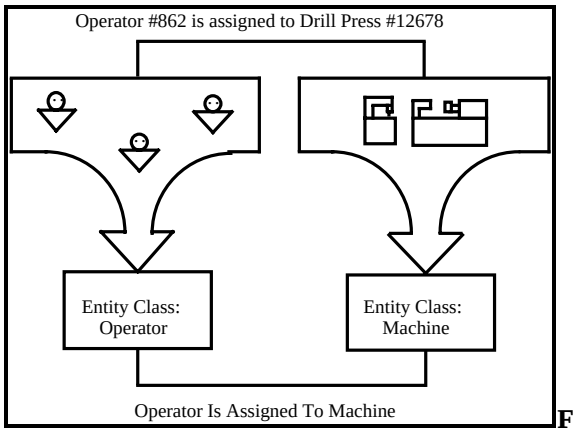


Figure 6-21. Relation Class Label

There are two aspects of the relation class which serve to describe the relationship which exists between the entities. They are the Relation Class Label and the Relation Class Ratio.

The relation class ratio serves to quantify (loosely) the number of relationships that may exist for each member of an entity class. It also defines the functional (existential) dependency that exists in the relationship. An existential dependency exists when the association between two entities is such that one cannot exist without the other existing first.

In this kind of relationship, the entity class, which is not dependent on the other one, is called the “independent” entity class; the entity class which is dependent on the other one is called the “dependent” entity class.

There are three expressions of relation ratio used in IDEF modeling:

1. One-to-One (1:1)
2. One-to-Many (1:N)
3. Many-to-Many (M:N)

These relation ratios have a specific syntax so that they can be expressed graphically. Figure 6-22 shows the syntactical representation of each of the relation ratio variations. A full diamond is used as the symbol representing a “zero, one or many” relationship, while a half-

diamond represents a “zero or one” relationship. Those relation classes where either of the entity classes can exist independent of the other one are referred to as “non-specific” relation classes; if either of the entity classes is dependent, the relation classes are referred to as “specific.”

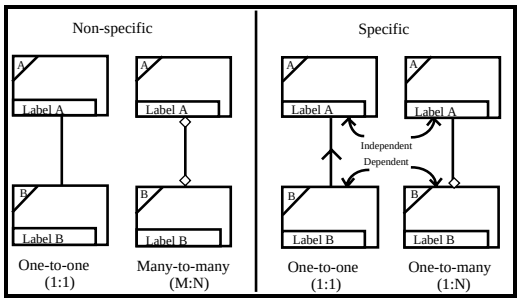


Figure 6-22. Relation Class Ratio Syntax

Learning to read and interpret these expressions and variations is one of the academic aspects of modeling. A closer look at the basic interpretation of this syntax will help you understand them better.

Using the same example, Figure 6-22, the basic interpretation would be:

1. The “Non-Specific” Relation Classes

   1:1 Each member of entity class “A” is related to zero or one member of entity class “B”; conversely, each member of entity class “B” is related to zero or one member of entity class “A.”

   M:N Each member of entity class “A” is related to zero, one, or many members of entity class “B”; conversely, each member of entity class “B” is related to zero, one, or many members of “A.”

2. The “Specific” Relation Classes

   1:1 Each member of entity class “A” is related to zero or one member of entity class “B”; conversely, each member of entity class “B” is related to exactly one member of entity class “A.”

   M:N Each member of entity class “A” is related to zero, one, or many members of entity class “B”; conversely, each member of entity class “B” is related to exactly one member of entity class “A.”

   Note: In the “specific” form of relation class expression, entity class “B” is existentially dependent on entity class “A”; that is, the member of entity class “A” to which the member(s) of entity class “B” is related, must exist “before” the related member or entity class “B.”

Once the relation class syntax has been selected, the modeler must select a label and develop a definition for the relation class. The relation class label is a short phrase typically verb-like, with a conjunction to the second entity mentioned (as a preposition introduces its phrase). This phrase reflects the meaning of the relationship represented. Often, the relation class label is simply a single verb, but verbs, adverbs and prepositions also make frequent appearances in relation class labels. Once a relation class label is selected, the modeler should be able to read the diagram and produce a meaningful sentence defining or describing the relationship between the two entity classes.

In the case of the "specific" relation class form, there is always an entity class and a entity class; the relation class label is interpreted from the "independent" end first, then from the dependent. This is illustrated in Figure 6-23.

In the case of the "non-specific" relation class form, there are two relation class labels, one for each entity class, separated by a "/" mark. In this case, the relation class labels are interpreted from "top to bottom" or "left to right" (depending on the relative positions of the entity classes on the diagram) and then in reverse. This is illustrated in Figure 6-24.

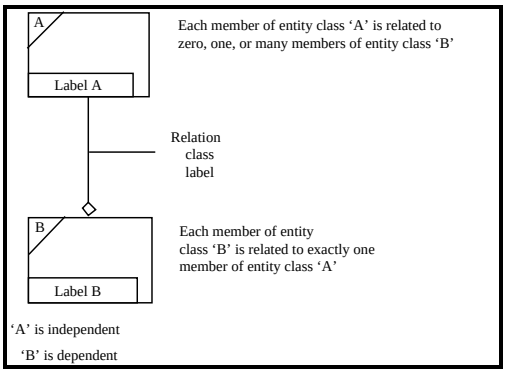


Figure 6-23. One-to-Many

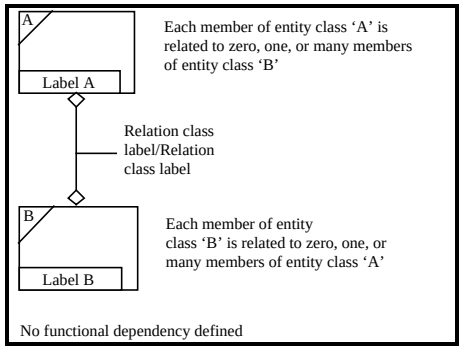


Figure 6-24. Many-to-Many

The emphasis on the relation class labels is that they must carry meaning. There must be some substance in what they express. The full meaning and the modeler's rationale in selecting a specific relation class label must be documented textually by what is called the Relation Class Definition. The relation class definition is a textual statement explaining the relation class meaning. The same rules of definition character apply to the relation class definition as applied to the entity class definitions; they must be:

1. Specific
2. Concise
3. Meaningful

6.4.1 Basic Process

The first step in Phase Two is to identify the relation classes that are observed between members of the various entity classes. This task requires the development of a Relation Matrix as shown in Figure 6-25. A Relation Matrix is merely a two-dimensional array, having a horizontal and a vertical axis. One set of predetermined factors (in this case all the entity classes) is recorded along one of the axes, while a second set of factors (in this case, also all entity classes) is recorded along the other. An "X" placed in the intersecting points along which any of the two axes meet is used to indicate a relationship which may exist between the entity classes involved. At this point, the nature of the relationship is unimportant; the fact that a relationship may exist is sufficient.

Once the Relation Matrix is completed, the modeler can start to produce rough drafts of the entity class diagrams. Basically, the entity class diagrams at this stage represent a simple translation from the Relation Matrix to IDEF1 diagram format.

One diagram is produced for each line, that is, each entity class represented on the horizontal axis of the matrix. Each of these entity classes is the subject that is depicted in the center of only one diagram. The entity classes to which they are related, as indicated by the "X"s in the vertical axes of the matrix, are drawn to the top, sides, and bottom of the subject entity class on the diagram. A sample "preliminary" or "rough draft" entity class diagram is depicted in Figure 6-26. It is important to note that at this stage there has been no attempt to define what kind of relationship is being reflected.

In the earlier material it was pointed out that there are two categories of relation classes: the non-specific and the specific. In this phase of information modeling, all of the relation class forms are legal for the modeler to use. The next step in preparing these preliminary diagrams involves selection of the appropriate relation class form to apply to the lines between entity classes. Once the proper symbols have been selected and drawn in on the rough draft, the modeler can choose a label for the relation class; one that aptly and succinctly describes the relationship represented. The modeler may choose to label some non-specific one-to-one (1:1) relation classes as "unknown" if more information is needed to understand the nature of the relationship represented.

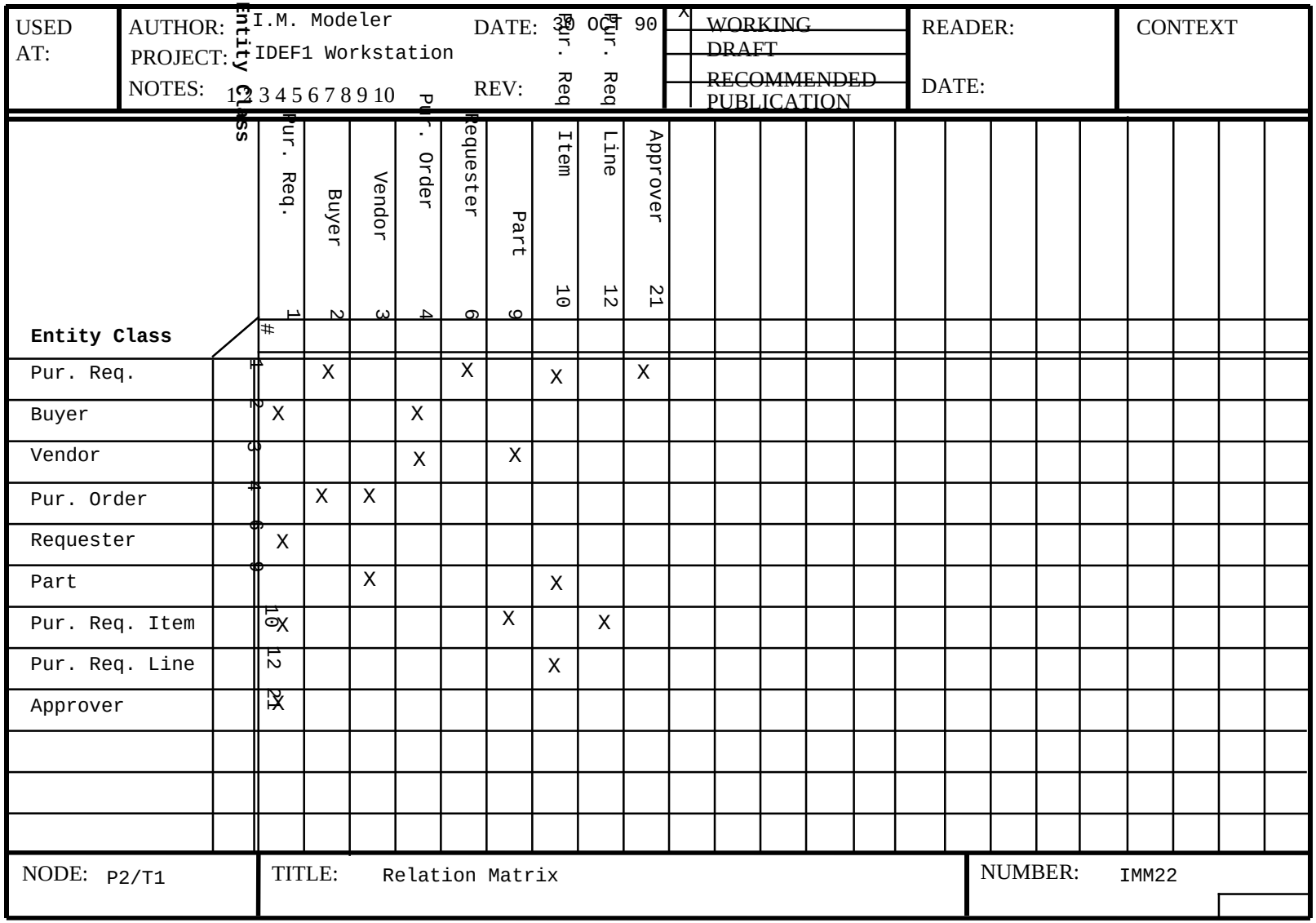


Figure 6-25. Relation Matrix

Now the modeler is ready to formalize the entity class diagram, based on the "rough drafts" that have been completed. To begin with, a standard entity class diagram format is used in the development of the formal entity class diagrams.

There are specific guidelines the modeler must follow with the formal entity class diagrams. These are:

1. The "subject" entity class will always appear in the approximate center of the page.

2. The "independent" entity classes should be placed above the subject entity class.

3. The "dependent" entity classes should be placed below the subject entity class.

4. The relation class lines radiate from the "subject" entity class box to the related entity classes. The only relationships shown on the diagram are those between the subject" entity class and the related entity classes.

5. Every relation class line has a label; in the case of non-specific relation classes, the line has two labels, separated by a "/."

6. The only exceptions to the basic formatting guidelines are the non-specific relation class forms, which are frequently shown to the side of the subject entity class box.

The modeler now must define the relation classes. This does not mean simply defining or restating the syntactical interpretation of the diagram. Rather, it involves defining the rationale behind the selection of the relation class syntax and relation class name. An example of several relation class labels and their definitions is shown in Figure 6-27.

Relation Class Label	Relation Class Definition
Is authorized to perform as	Selected employees are authorized to perform as buyers, and are provided unique identification pertaining to this role, distinct from their employee identification.
Is Currently assigned to	Each operator may be assigned to some number of work stations during any shift, but this relationship reflects the one the operator is assigned to at the moment.

Figure 6-27. Relation Class Definition

It is important to keep in mind that the relation class definition must have meaning too. A clear and precise definition will provide for a clear and precise understanding by the reviewers. Relation class definitions must leave little room for doubt or confusion in order to fulfill their intended purpose.

Now the modeler is ready to draft the node cross-reference statements and forms. The node cross-reference is basically a restatement of the entity class diagram. It is through this medium that interpretation of the diagram syntax is specifically illustrated. A sample of the node cross-reference sheet is illustrated in Figure 6-28. The upper left-hand corner of one of the fields for entity class names on each line is partially filled in, i.e. darkened in a triangular figure. This indicates the starting point for interpreting the relationship illustrated on the diagram. In Figure 6-28, for example, line number 1 would read, “Customer authorizes contract spec. change.” Line number 2 would read, “Contract spec. change results in engineering change.”

USED AT:	AUTHOR: I.M. Modeler PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 30 OCT 90 REV:	WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

#	SUBJECT ENTITY CLASS LABEL	RELATED ENTITY CLASS LABEL	#	RELATION CLASS LABEL
84	Contract Spec. Change	Customer	P	Authorizes
84	Contract Spec. Change	Engineering Change	14	Results In

NODE: P2/E84 (x1)	TITLE: Entity Class Diagram Node Cross-Reference: Contract Specification Change	NUMBER: IMM488

Figure 6-28. Node Cross-References Sheet

Please note that no indication of the ratio of occurrence is indicated on the node cross-

reference. This is simply an aid in double checking the interpretation of the relation class and diagram structure. The sentence structure to interpret the non-specific relation class must be performed in two directions, one for each of the non-specific ends of the relationships.

At this point the material available for each entity class set includes:

1. The entity class definition.
2. The entity class diagram.
3. The relation class definitions.
4. The near cross-reference.

The information about an entity class can be expanded by the addition of reference diagrams, at the modeler's discretion.

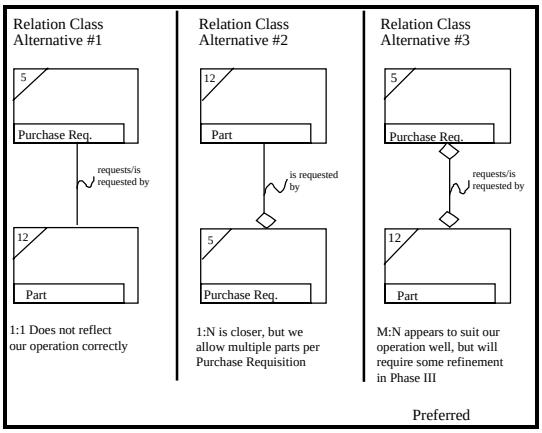


Figure 6-29. Reference Diagram

Reference diagrams (diagrams For Exposition Only, sometimes called "FEOs") are an optional feature available to the modeler, to which individual modeler conventions may be applied. These diagrams are primarily platforms for discussion between the modeler and the reviewer(s). They offer a unique capability to the modeler to document rationale, discuss problems, analyze alternatives, and look into any of the various aspects of model development. One example of a reference diagram is shown in Figure 6-29. This figure depicts the alternatives available in the selection of a relation class and is marked with the modeler's preference.

Another type of reference diagram, illustrated by Figure 6-30 depicts a problem confronted by the modeler. In this example, the modeler has identified the problem and its complexities for the reviewer's attention.

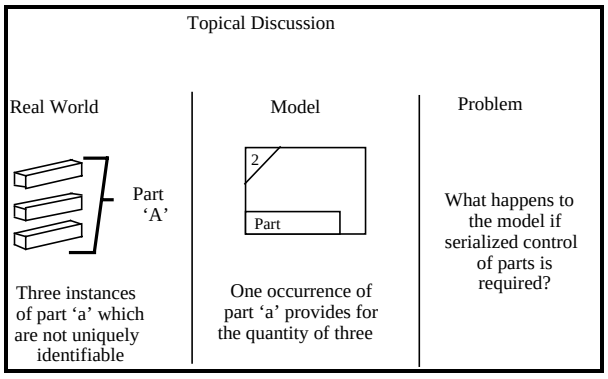


Figure 6-30. The "FEO" - A Reference Diagram

By this stage, the modeler has compiled sufficient information to begin the development and distribution of Phase Two kits.

6.4.2 Phase Two Kits

The kit structuring in Phase Two is somewhat different than kit structuring in Phase One because it is more complex. An example of how a Phase Two kit is structured is reflected in Figure 6-31.

A kit cover sheet must be prepared for each kit. lt must denote the content of the kit (lower left-hand corner) as well as other pertinent information. A sample is shown in Figure 6-32.

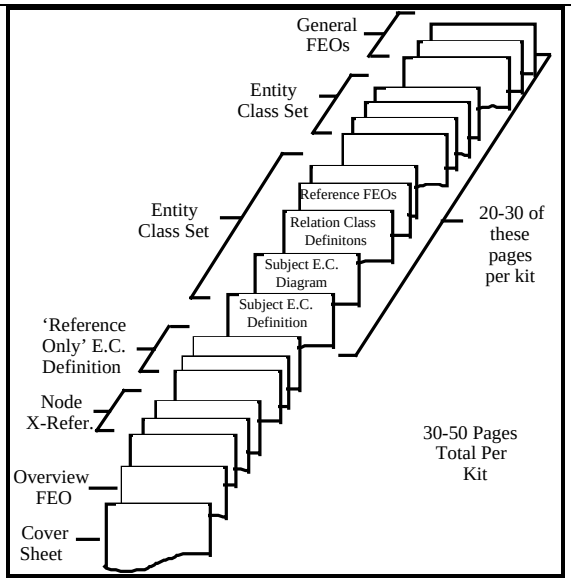


Figure 6-31. Phase Two Kit Structure

Once the specific entity classes to be included in a kit have been selected, a reference diagram must be constructed which reflects the relationships between the "subject" entity classes. This diagram is called the kit overview diagram. In some cases, all entity classes in the kit will be directly related with one another, but, in others, the entity classes may be only indirectly related, usually through other entity classes which are not included in the kit. These indirect relationships are customarily reflected by enclosing the non-subject entity class(es) within a large dotted line box and indicating textually that they are not included as subject entity classes in the kit. This is illustrated in Figure 6-33. The kit overview diagrams (included in kits as FEOs) are designed to:

1. Reflect the entity classes which are "subjects" of the kit.
2. Inform the reader of what the general content of the kit is.
3. Show the relationship which exists between the "subjects" in the kit.

The specific content of the kit is primarily a function of the number of pages associated in total with each entity class set and to a certain degree the level of complexity of the entity class sets. The primary objective of the modeler at this point is to constrain the kits to a point where no more than 1 1/2 to 2 hours of the reviewer's time (maximum) will be occupied in reviewing and commenting on any single kit. The general Phase Two kit sizing parameters to meet are:

1. 30-50 total printed pages per kit.

2. 4-6 entity class sets per kit.

3. 1 1/2-2 hours of review time per kit per reviewer.

There are typically four to six pages of information per entity class set in Phase Two. This statistic, coupled with two other aspects of kit structuring, will help in determining an equitable kit sizing:

1. The node cross-reference pages for the "subject" entity classes in the kit are extracted from the entity class sets and placed in a group toward the front of the kit. These form an index of the contents of the kit.

2. The entity class definitions for the "subject" entity classes are included with the entity class sets. The entity class definitions for the non-subject (related) entity classes shown in the node cross-reference are grouped near the front of the kit, to be used for reference only.

One of the efforts that consumes a lot of reviewer time is reading and commenting on the pages where it is not really necessary. It is a good practice for the modeler to draw attention to the reason for placing a page in the kit. Entity classes which are being released for the first time should be clearly pointed out. On the other hand, if a page is inserted simply for reference, then it should be clearly marked "For Reference Only" to avoid the reviewer spending a lot of time commenting on this page. The "informal" page markings help the reviewer focus attention on the appropriate aspects of the kit.

One important point to remember about the expansion which could occur in the Phase Two process is that any new entity classes entered into the model at this point must meet all the prior requirements of Phase One.

Whenever this occurs, the modeler is required to update the entity class pool and to proceed with the development of a new entity class definition, before the entity class(es) can be used in Phase Two material.

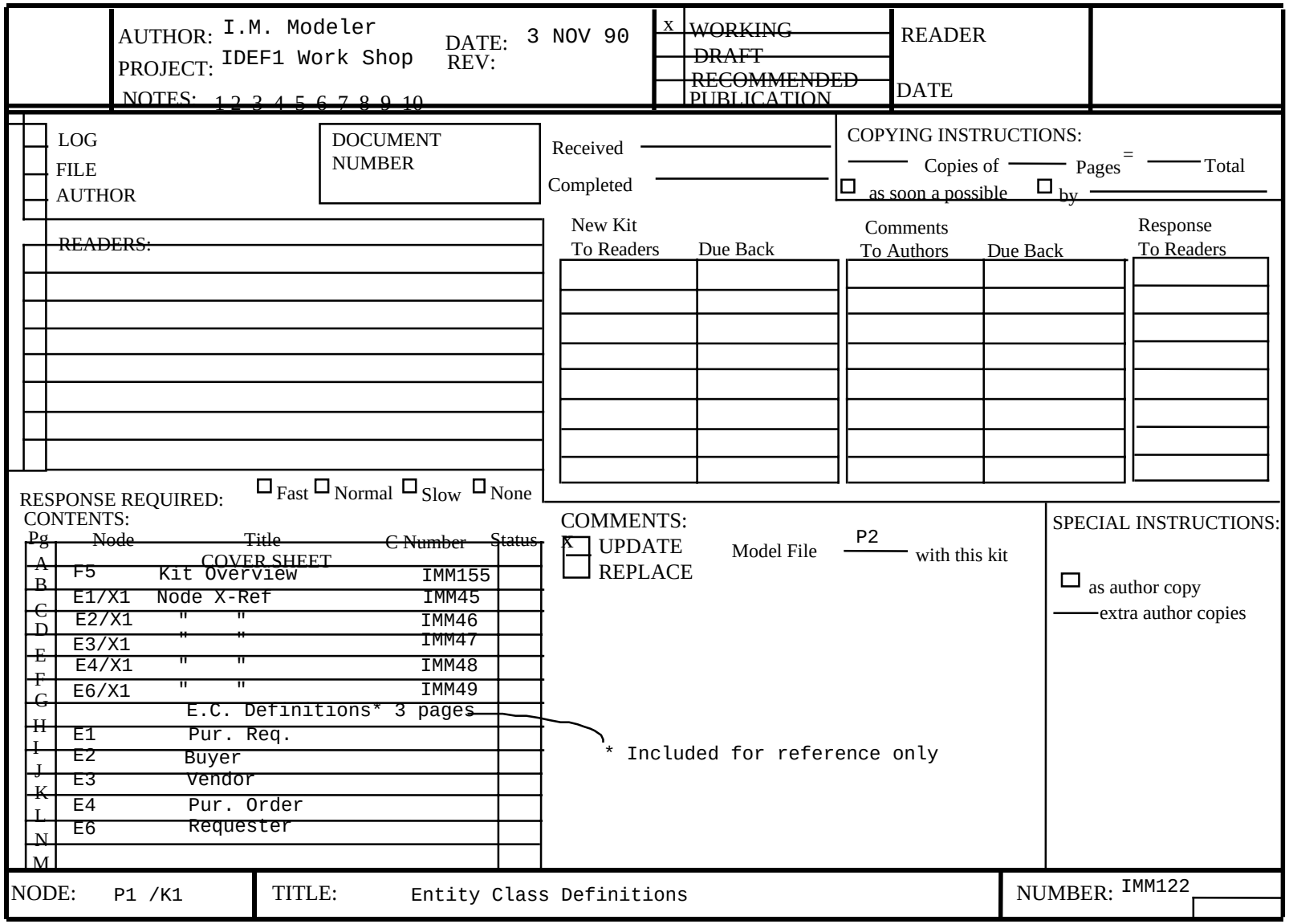


Figure 6-32. Phase Two Cover Sheet

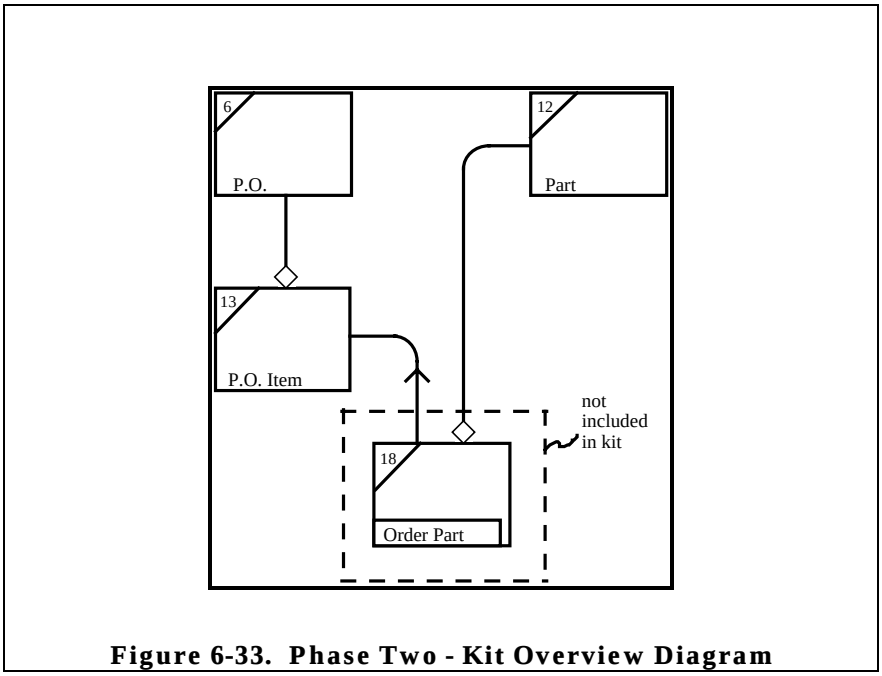


Figure 6-33. Phase Two - Kit Overview Diagram

6.5 Phase Three – Key Class Definitions

Phase Three of the IDEF1 methodology deals with the identification and definition of an element of information about entities referred to here as Key Classes. The purpose of the Phase Three activity is to identify those attributes by which each entity in an entity class can be, uniquely identified. Phase Three expands upon the work done in Phase One and Phase Two. This will be accomplished through the attaining of the following objectives:

1. Refine the "non-specific" relation classes carried over from the Phase Two activity.

2. Identify the attribute classes readily observed within the context of the model.

3. Identify the key classes. These are the identifiers of entities by which they are uniquely identifiable one from the other.

4. Define the key attribute classes. These are the attribute classes which are used in key classes.

5. Construct the attribute class diagrams. These are extensions of entity class diagrams.

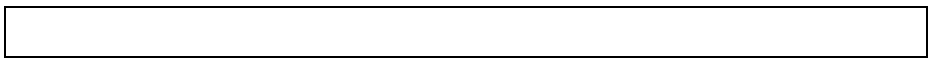
There are a significant number of products which result from Phase Three. These products include:

1. Attribute class pool
2. Attribute class diagrams
3. Key Class identifications
4. Entity class/attribute class matrix
5. Key attribute class definitions
6. Inherited attribute class cross-reference
7. Attribute class migration index
8. Refinement alternative diagrams
9. Function view diagrams
10. Entity class/function view matrix
11. Phase Three kits

Phase Three involves an extensive amount of product and a large number of activities. The basic terms which are used in Phase Three areas follows:

1. Attribute
2. Attribute Class
3. Key Attribute Class
4. Non-Key Attribute Class
5. Key Class
6. Key Class Migration
7. Inherited (Shared) Attribute Class

The basic element of IDEF1 diagramming is the entity class box, but in Phase Three it begins to take on characteristics which are different from those which have become familiar in Phase Two. The primary difference is the inclusion of certain attribute classes in the "subject" entity class box. An illustration of the Phase Three entity class box is shown in Figure 6-34.



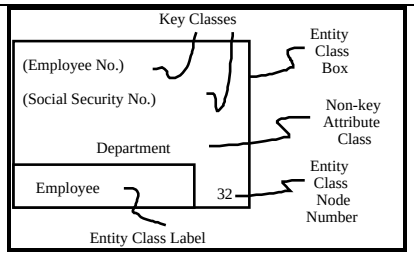


Figure 6-34. Phase Three Entity Class Box

It is important at this point that the definition and the meaning of the terms attribute and "Attribute Class" be reemphasized. An attribute is a property or characteristic of an entity. Attributes are composed of a name and a value. In other words, an attribute is one element of information that is known about a particular entity. Attributes are descriptors; they tend to be adjective-like in nature.

An example of some attributes and their respective entities, is shown in Figure 6-35. Note that the first entity, or individual, is identified with an employer number of "1" and the name associated with the entity is "Smith;" and that the job of the entity is "operator." These attributes, taken all together, describe the entity uniquely and separate that entity from other similar entities. Every attribute has both a name and a value. It is the unique combination of attribute values which describes a specific entity.

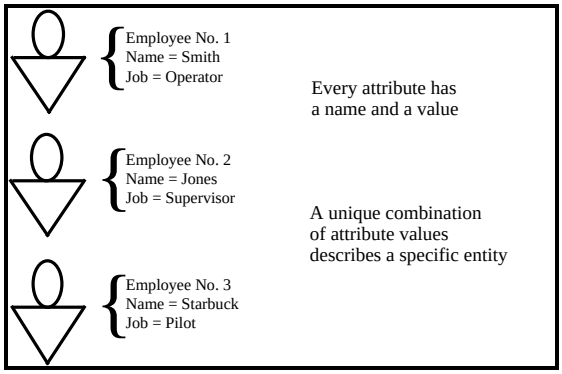


Figure 6-35. Attribute Examples

An attribute class represents a collection of attributes of the same name which apply to all the entities of the same entity class. Attribute class names are typically singular

"descriptive" nouns. In the example of the entity class "employee," there are several attribute classes including:

1. Employee number
2. Employee name
3. Employee job/position

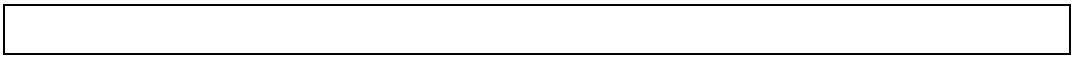
An example of how attributes are represented as attribute classes is shown in Figure 6-36. The attribute values belong to the entities. But the attribute classes themselves belong to the entity class. Thus, an "ownership" association is established between an entity class and some number of attribute classes.

An attribute class has only one owner. An owner is the entity class in which the attribute class originates. In our example, the owner of the attribute class "employee number" would be the entity class "employee." Although attribute classes have only one owner, the owner can "share" the attribute class with other entity classes. How this works will be discussed in detail in later segments.

An attribute class represents the use of an attribute to describe a specific property of a specific entity. Additionally, some attribute classes represent the use of an attribute to help uniquely identify a specific entity. These are informally referred to as key attribute classes.

Phase Three focuses on the identification of the "key" attribute classes within the context of our model. In Phase Four the "non-key" attribute classes will be identified and defined.

One or more key attribute classes form the "Key Class" of an Entity Class. A Key Class is defined as one or more key attribute classes used to represent the attributes required to uniquely identify each member (each entity) of an entity class. An employee number is an example of one attribute class being used as a Key Class of an entity class. Each employee is identified from all the other employees by an employee number. Therefore, employee number is the key class which we can say uniquely identifies each member of the entity class employee.



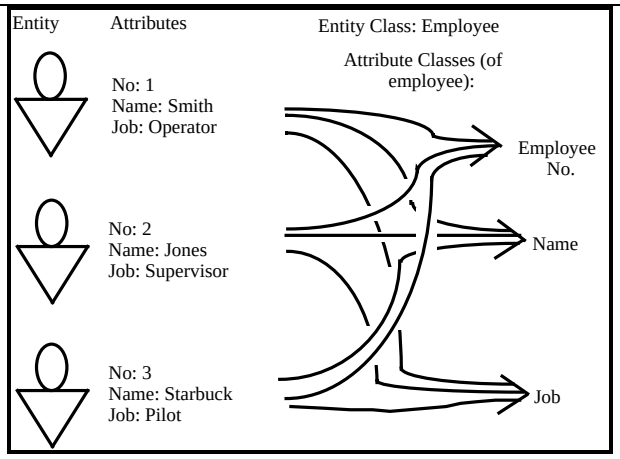


Figure 6-36. Attribute Classes Example

Key classes are included within the entity class box of the subject entity class on attribute class diagrams. Key classes are always underscored. There are several "visual" forms of key classes. The first of these is informally called the "simple" form. A "simple" Key Class is composed of a single "key" attribute class. A second visual form that a Key Class can take is formally called the "compound" form. A "compound" Key Class is made up of more than one "key" attribute class. Each of the attribute classes which make up a "compound" Key Class is separated by commas, but the entire Key Class is still underscored. A third visual form which a Key Class can assume is called "alternate" or "equivalent" Key Class. It must be emphasized that alternate key classes are exactly and precisely equivalent to their counterpart. That is, either Key Class will result in the same unique identification of precisely the same entity (member of the entity class). Alternate key classes may be either simple or compound. In either event, each alternate Key Class is parenthetically enclosed and underscored within the parentheses. Attribute classes which are used as part of a Key Class are often referred to as Key Attribute Classes or Key Class Members. An example of the various Key Class forms is shown in Figure 6-37.

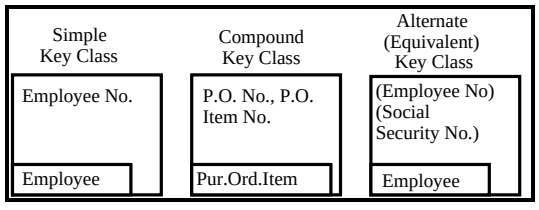


Figure 6-37. Key Class Forms

Earlier, it was mentioned that the entity class which "owns" an attribute class can "share" it with another entity class. Attribute classes become shared through the process of Key Class migration. Key Class "migration" is when "key" attribute classes are moved from one entity class to another. The rules which govern Key Class migration are as follows:

1. Migration always occurs from the independent to the dependent entity class in the related pair.

2. The entire Key Class (that is, all attribute classes which are members of the Key Class) must migrate once for each relation class shared by the entity class pair.

3. Non-key attribute classes never migrate.

Attribute classes that migrate from one entity class to another are called "Inherited" or "Shared" . An inherited attribute class must be a key attribute class (a member of a Key Class) of the entity class from which it migrated, but an inherited attribute class does not always have to be a member of the Key Class of the entity class to which it migrated (by which it is inherited). All attribute classes are either owned or inherited by the entity class with which they are associated.

An example of the migration of an attribute class from one (independent) entity class to another (dependent) entity class is shown in Figure 6-38. In this example, purchase order number (the Key Class of the entity class purchase order) migrates to (is inherited by) the entity class purchase order item. It is then used by purchase order item as a member of its Key Class in conjunction with another attribute class "owned" by purchase order item, which is called purchase order item number. The two attribute classes (purchase order number and purchase order item number) form the Key Class for the entity class purchase order item.

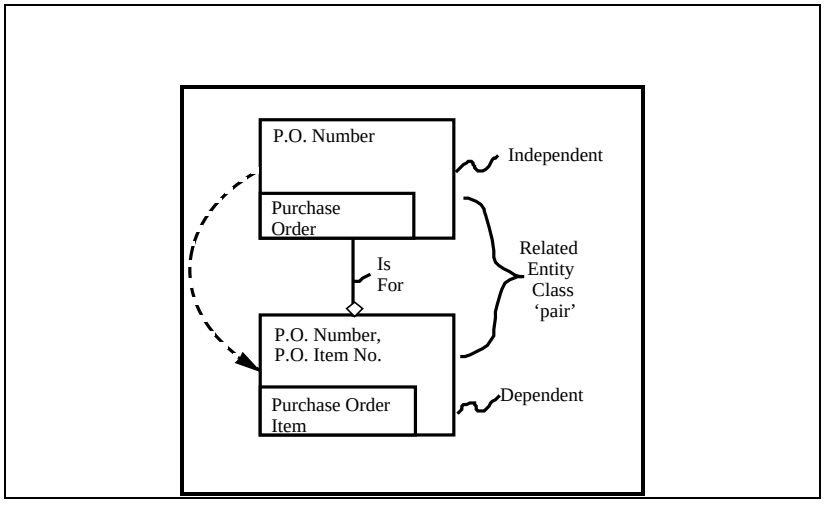


Figure 6-38. Key Class Migration

The process of Key Class migration is one of the techniques used to drive the model in total to the next lower level of detail. Before Key Class migration can be accomplished, the modeler must first resolve all non-specific relation classes carried over from Phase Two. This resolution causes the stabilization of the functional (existential) dependencies within the model. You may recall from our Phase Two discussion that functional dependency is reflected only in the specific relation class syntax.

6.5.1 Phase Three Process

Phase Three of the IDEF1 information modeling technique is composed of the following processes:

1. Refine "non-specific" relation classes.
2. Construct function view diagrams.
3. Initiate construction of attribute class pool.
4. Identify key classes.
5. Define key attribute classes.
6. Formalize Phase Three entity class sets.
7. Build and distribute Phase Three kits.

The first step in this phase is to insure that all non-specific relation classes observed in Phase Two have been refined. Phase Three requires that only a "specific" relation class form be used; either the specific one-to-one or specific one-to-many. Figure 6-39 illustrates the "specific" form. To meet this requirement, the modeler will employ the use of "refinement" alternatives. A alternative diagram is a type of reference diagram. They are normally divided into two parts: the left part deals with the subject (the "non-specific" relation class to be refined), while the right part deals with the refinement alternative. An example of a refinement alternative dealing with a many-to-many resolution is exhibited in Figure 6-40.

The process of refinement of relation classes translates or converts the non-specific relation classes into some number of "specific" relation classes. Out of this process new entity classes frequently evolve. It is in Phase Three that we now begin to see a new kind of entity class.

In all earlier phases we've been working with what we might informally call the "natural" entity classes. A natural entity class is one which we will probably see evidenced in the source data list or in the source material log. A "natural" entity class would include such names as:

1. Purchase order
2. Employee
3. Buyer

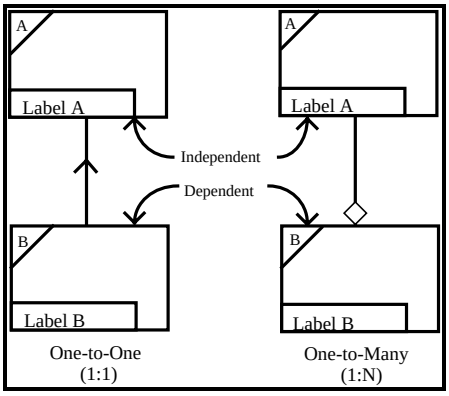


Figure 6-39. Specific Relation Classes

It is now during Phase Three that we begin to see the appearance of what may informally be called the entity class. Figure 6-41 is an example of a "derived" entity class. In this example, a "derived" entity class is used to represent the ways in which purchase requisition items are related to purchase order items. Notice that a derived entity class name label is somewhat different in character than the object nouns typifying the natural entity classes. It is also not unusual to tag the entity class both of a derived entity class with an entity class description to clarify the reason for the existence of the entity class. One of the subtle differences between the natural and derived entity classes is in the entity class names. Typically the entity class name for natural entity classes is a singular common noun. On the other hand, the entity class name of the derived entity classes is often an artificial type of noun.

The derived entity class is somewhat more abstract in nature and normally results from the application of rules governing the validity of entity classes which are first applied in Phase

Three. The first of these rules is the rule requiring refinement of all "non-specific" relation classes. This process of refinement is the first major step in stabilizing the integrated information structure.

This process of refinement involves a number of basic steps, including:

1. The production of one or more rough draft refinement alternatives for each "non-specific" relation class.

2. The selection by the modeler of a preferred alternative which will be reflected in the Phase Three diagrams.

3. The updating of Phase One information to accommodate inclusion of new entity classes resulting from the refinement.

4. The updating of Phase Two information to accommodate utilization of new entity classes resulting from the refinement.

Once refinement is completed, the modeler can begin to determine the "functionality" of the model.

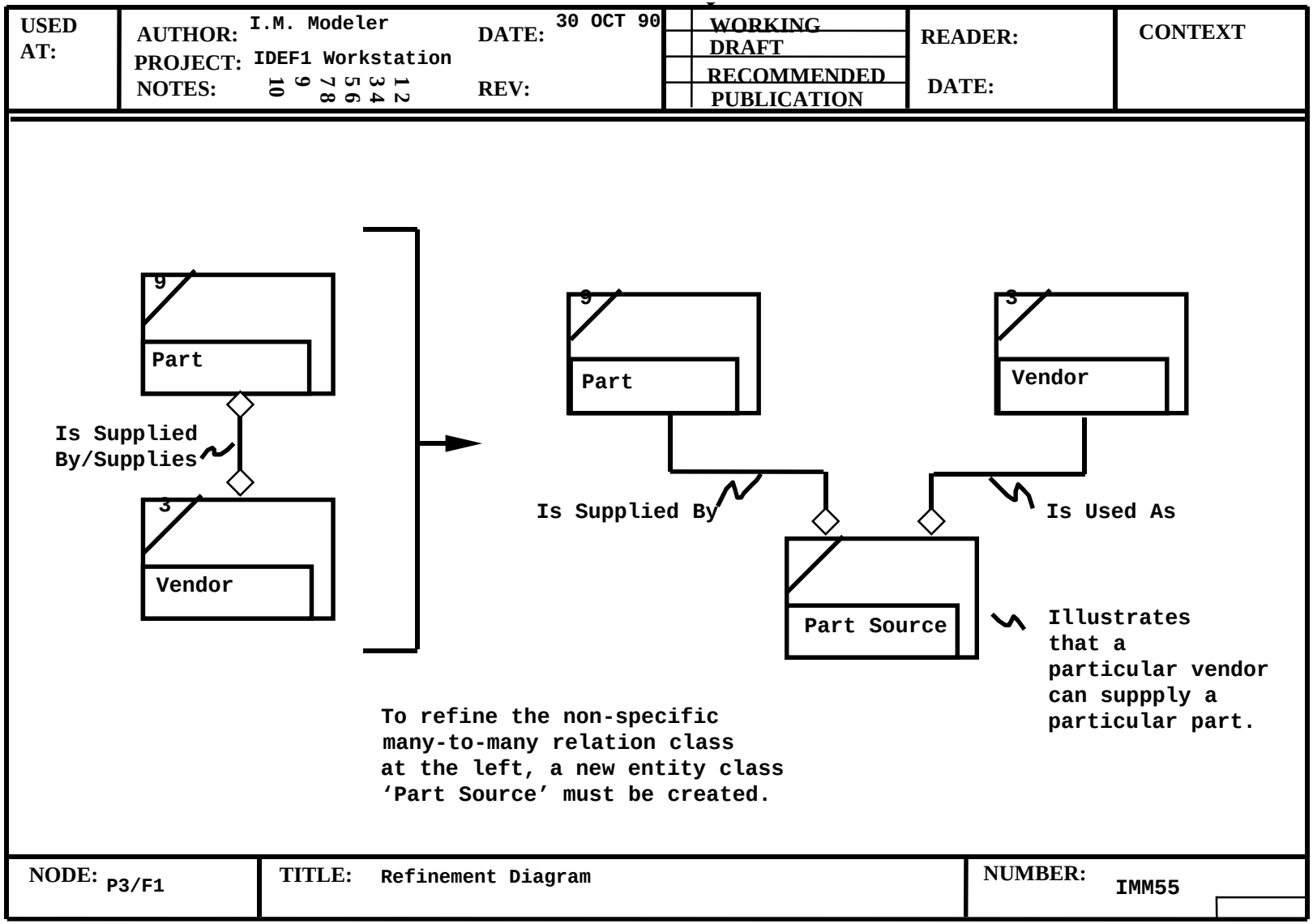


Figure 6-40. Refinement Diagram

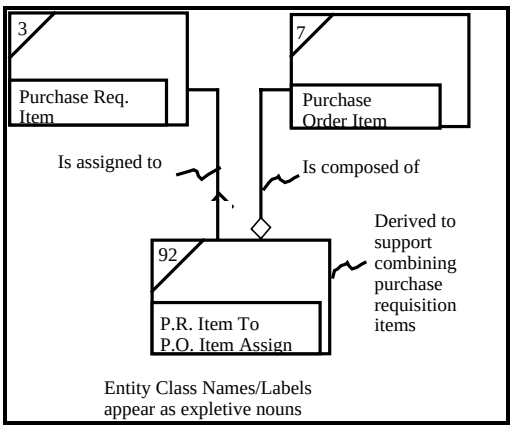


Figure 6-41. The "Derived" Entity Class

6.5.2 Function View FEOs

Typically, the volume and complexity level of the information model at this point is becoming appreciable. It was quite natural during Phase One to evaluate each entity class independently of the other entity classes, because the entity classes were simply definitions of words. In Phase Two, it is practical to continue evaluating one entity class at a time, because the total volume of entity classes is usually not so large as to prohibit the development of a mental image of the whole model at one time. In Phase Three, the volume of entity classes and the complexity of relationships being reflected in the model is such that an individual can no longer retain a total mental image of the meaning of the model. The model must be reviewed and validated from a new perspective. This perspective enables the evaluation of the model in a fashion more directly related to the "functional" aspects of the enterprise being modeled. This perspective is represented by what is called the "function view." A function view is a general diagram For Exposition Only (FEO). Its purpose is to establish some limited context within which portions of the model can be evaluated at one sitting. The primary characteristics of a function view are as follows:

1. A function view deals with a single topic. The topic might be a document, a report, or a process.

   What the function view reflects is the cluster of entity classes and relation classes which represent the information structure of the document, report, or process.

2. Typically, a function view is limited to 25 to 30 entity classes. This is simply a practical limitation established to facilitate the development and maintenance of a mental image of the topic and related information structure on the part of the reader.

The reason for the Function View Diagram is to focus attention on particular aspects of the information model. By time the Phase Three refinement is nearing completion, an information model is a relatively complex object. Because of this complexity different images will be perceived in the model by different individuals. The model at this stage might be likened to a collage of lines and images. The interpretation of the perceiver of the model will vary somewhat based on the perspective. This variance in perception is like the perception which takes place in an "ink blot" examination, or some similar type exercise. In a model which is being utilized to facilitate requirements definition, though, wide variances of perception are intolerable.

Figure 6-42 depicts an excellent example of this type of perception problem that can produce an extreme difference of opinion.

Which figure, do you see first? The old woman? Or the young woman? Have you now seen them both?

What could have helped you find them more quickly? More easily?

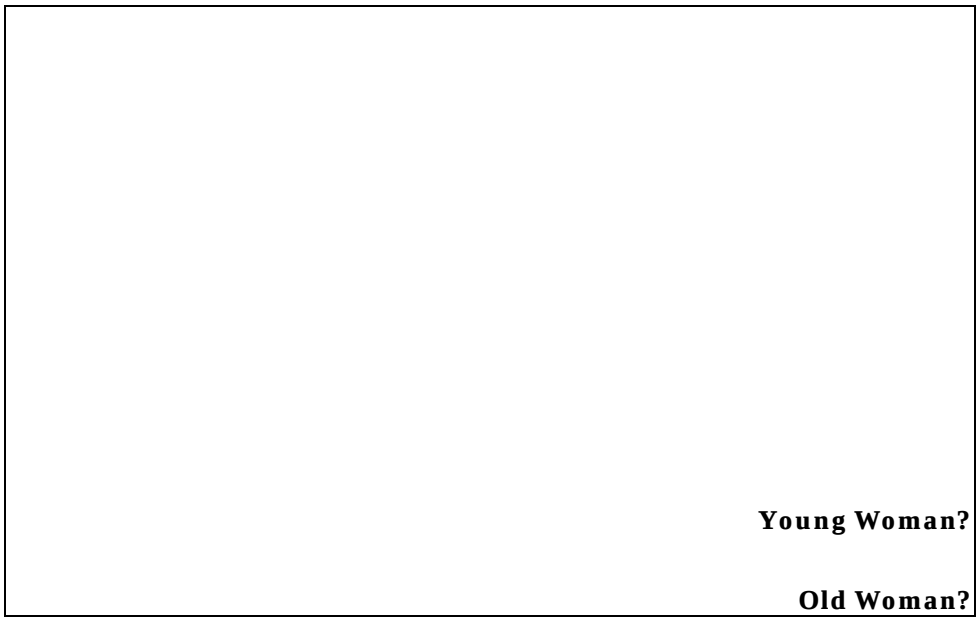


Figure 6-42. Perspective

One of the tools by which perspectives of the information model can be examined is through the use of function view diagrams. These are diagrams which emphasize specific characteristics or features of the perspective. The connections, key relationships, etc., are used to draw attention onto the proper perspective. An analogy can be drawn from the "old woman/young woman" example.

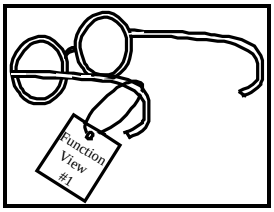


Figure 6-43. Perspective Number One

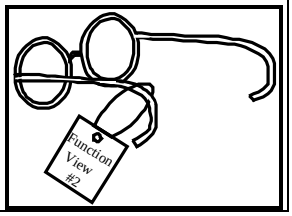


Figure 6-44. Perspective Number Two

If it were possible to develop a set of glasses which would "see" only the aspects of one image or the other, the ability to distinguish between the two would be much keener. Those glasses could be referred to as the "function views," and one pair would be required for each view (perspective). Figure 6-43 illustrates "perspective" number one, Figure 6-44 "perspective" number two.

The function view is very much like colored, "perspective sensitive" glasses. Each function view causes the focus of attention to be on a particular subset of the information model and causes a particular image (the image of the topic) to appear and other images to go away. This does not change the content of the model. It simply adjusts the focus of the reviewers to a particular perspective.

Obviously, function views can be instrumental in the evaluation and validation of the information model. Just as obviously, the modeler must exercise some care in the determination or selection of topics illustrated in a function view. Two methods which have been used with some success are:

1. Select sample source material to use as the topic of a function view. For example, a purchase order, etc.

2. Relate the function view to job categories, or specific processes, represented by the administrative groups or functional areas identified as sources in Phase Zero.

A Function View Diagram is basically a type of reference diagram. It is For Exposition Only– a FEO. Function View Diagrams represent a single topic on which they focus. They may involve, however, some 25 to 30 entity classes. Each of the entity classes and selected relation classes which are included in the function view are there because they contribute to the topic, which is the subject of the function view. Anything that does not directly contribute to the topic is de-emphasized as "transparent" to the function view. The function view then reflects some portion of the existing model. It must be emphasized that anything within the context of the function view (any entity classes, relation classes, key classes, etc.) must be in the model proper. If, in the process of constructing a function view some entity

classes are discovered (more than likely, derived), they must be then added to the model, starting with the entity class pool and proceeding on through the modeling activities.

The purpose of the function view is to represent some clustering of entity classes and relation classes. This information may be that which is:

1. Required to reconstruct a document (perhaps some source material).

2. Required to reconstruct information used in, by, or produced by a process (perhaps something which has been observed and documented).

In Figure 6-45 for example, the information within the sample function view can be used to reconstruct a purchase order, or to reconstruct a report about some number of purchase orders. When constructing a function view, the author must have in mind the topic, so that it can be precisely expressed.

The information which can be reconstructed using the function views may not always be that kind of information commonly found in paper form. In fact, it may be information which is most often used in the form of an inquiry, such as when trying to:

1. Determine where something is stocked.

2. Determine where something can be stocked.

Figure 6-46 is in example of a function view which might deal with such a topic. This function view perhaps relates more to information used in a process than it does information used in a document and it is extremely important that the modeler be precise about the intent of the function view. This involves developing a function view description which is a textual definition of the topic of the function view. It should highlight both the purpose of the function view and the viewpoint of the function view. The reviewer needs to know and to understand what the cluster of entity classes is intended to represent and to whom it would be important. An additional area which should be included in the function view description is some textual or graphic definition of how this function view relates to the scope of the model as defined in Phase Zero.

It would seem feasible to represent the entire information model at some point in a series of interlocking function views, somewhat like putting together a jigsaw puzzle. Picking up one piece of the puzzle and looking at it, without an understanding of the total context within which the piece fits may make it somewhat difficult to relate to.

The point being illustrated here is that in order to optimize the understanding of the function view, the modeler must precisely define how he intends the function view to be used and by whom. That way reviewers with varying perspectives and purposes can properly relate to the viewpoint projected, and comment intelligently.

A function view is used to focus attention on a single topic and how that topic fits into the model. It helps the reader relate to the detail of the model within the context of the defined scope.

An added value of the function views is that the process of constructing them, the modeler frequently discovers the need for additional "derived" entity classes to express the meaning of information being represented. This aspect of the function view makes it a practical aid in the Phase Three refinement process.

6.5.3 Attribute Class Pool

The next step in the Phase Three process is to initiate construction of the attribute class pool. An attribute class pool is very similar to the entity class pool except that it is a collection of potentially viable attribute class names. Each name in the attribute class pool occurs only once and each is assigned a unique identifying number.

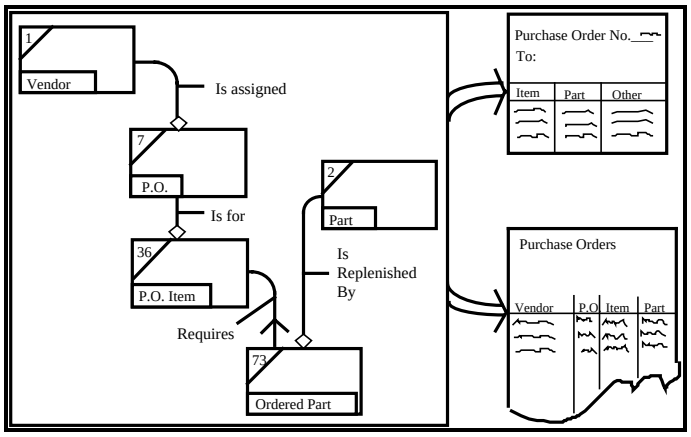


Figure 6-45. Purpose of the Function View

The process of constructing the Phase Three attribute class pool is similar in nature to construction of the entity class pool as well. In Phase One, when constructing the entity class pool, we extracted names from the Phase Zero source data list which appeared to be object nouns. Now we will return to the source data list and extract those names which appear to be “descriptive” nouns. Descriptive nouns (nouns which are used to describe objects) most commonly represent an attribute class.

Figure 6-47 reflects one page of an attribute class pool. Note that each attribute class name occurs only once in the pool and that each attribute class name has a unique identifying number.

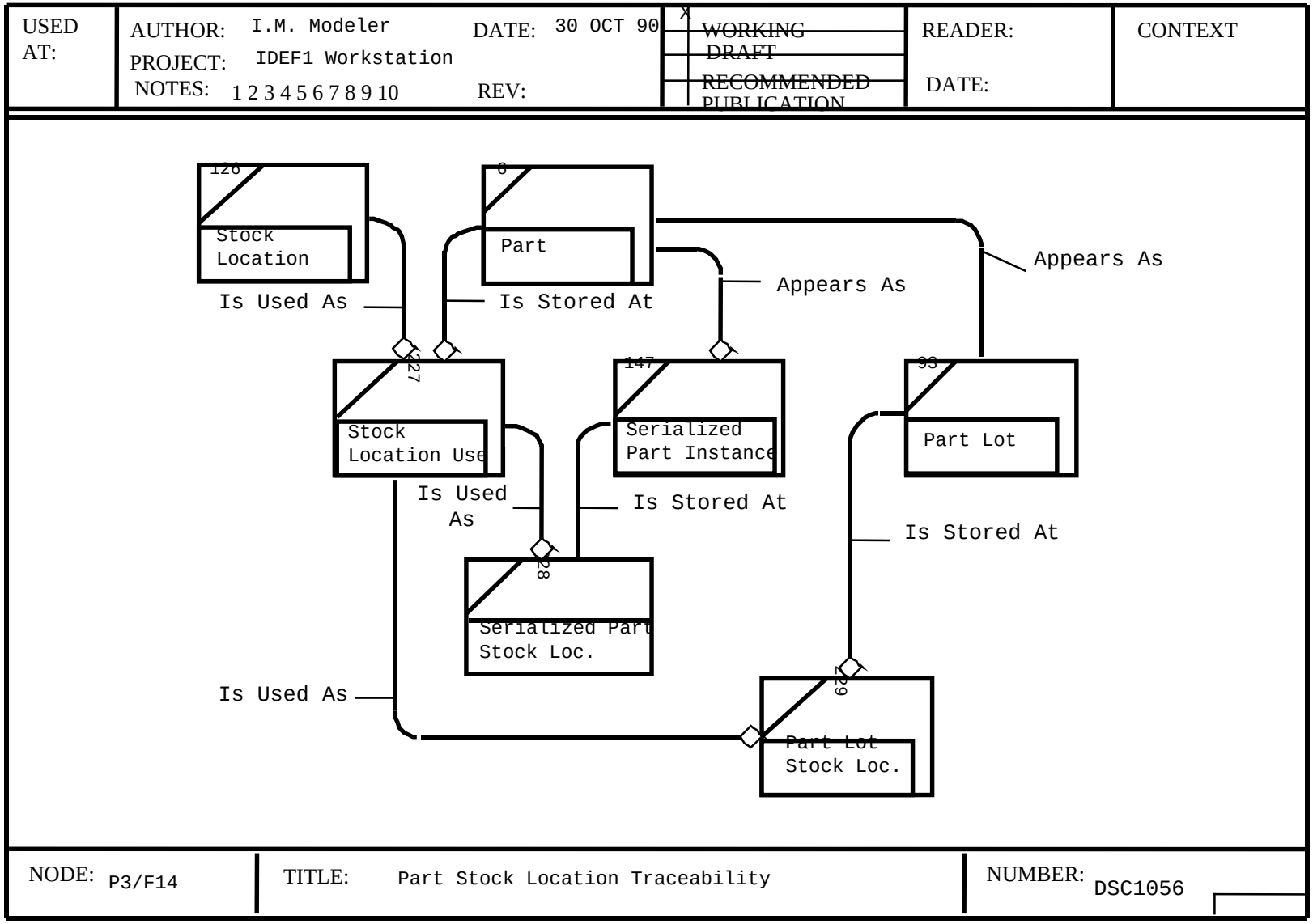


Figure 6-46. Function View Example

Many of the names on the Source Data List from Phase Zero were entered into the entity class pool in Phase One as potential entity classes, but some may have been recognized in Phase Three as not qualifying as entity classes. In all probability, these were attribute classes and many names which were not selected from the list in the first place were probably attribute classes. The list, in conjunction with the knowledge gained during Phase One and Phase Two, is the basis for establishment of the attribute class pool. The attribute class pool is a list of potentially viable attribute classes observed within the context of the model. This list will be appreciably larger than the entity class pool.

The attribute class pool is the source of attribute class names which are used in the model. In the event that attribute classes should be discovered in later phases of the modeling effort, these attribute classes must always be added to the attribute class pool, assigned a unique identifying number and progress from there to their intended use in the model.

In Phase Three, construction of the attribute class pool is initiated with the entry of Attribute Classes used to identify entity classes; i.e., attribute classes which are members of key classes.

6.5.4 Identifying Key Classes

The identification of key classes (the selection of attribute classes as members of a Key Class) starts with an evaluation of the way in which attributes are used in the enterprise being modeled. This may require that the modeler be able to trace a given attribute class back to the original source material, since that is most commonly the location where the use of an attribute class is best represented.

USED AT:	AUTHOR: I.M. Modeler DATE: 31 OCT 90 PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

Attribute Class			Attribute Class		
I.D. No.	Name	Source Data ID #	I.D. No.	Name	Source Data ID #
A1		SD1	A18	Quality Control Approval Code	SD17
A2	Buyer Code	SD2	A19	Taxable Code	SD19
A3	Vendor Number	SD3	A20	Resale Code	SD20
A4	Order Code	SD4	A21	Pattern Number	SD21
A5	Change Number	SD5	A22	Payment Terms	SD22
A6	Ship To Location	SD6	A23	Freight On Board Delivery Location	SD18
A7	Vendor Name	SD8	A24	Purchase Requisition Item Number	SD23
A8	Vendor Address	SD5	A25	Quantity Ordered	SD24
A9	Confirmation Code	SD9	A26	Quantity Unit of Measure	SD25
A10	Confirmer's Name	SD9	A27	Part Number	SD26
A11	Extra Copy Code	SD10	A28	Part Description	SD27
A12	Requester Name	SD11,42	A29	Unit Price	SD28
A13	Department Code	SD12	A30	Price Unit of Measure	SD29
A14	Ship Via	SD13	A31	Requested Delivery Date Purchase Requisition Line Code	SD31
A15	Buyer Name	SD14	A32	Requested Delivery Quantity	SD32
A16	Purchase Order Number	SD15	A33		SD33
A17	Purchase Requisition Issue Date	SD16	A34	Commodity Code	SD30

NODE: P3/X1	TITLE: Attribute Class Pool	NUMBER: IMM55

Figure 6-47. Attribute Class Pool

The first step in identifying key classes is to select the entity classes which have no dependent relationship. That is to say those entity classes which are not existentially dependent on any other entity classes. These generally represent the entity classes whose key classes are most obvious. Figure 6-48 illustrates this process.

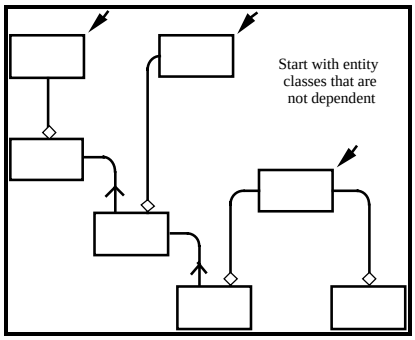


Figure 6-48. Key Class Identification

Some number of the attribute classes will be selected or identified as members of key classes somewhere within the model. Not all attribute classes will be used as key classes. Those which are not used as key classes in Phase Three will be dealt with in Phase Four.

The reason for selecting those entity classes which are wholly independent in the model is that the key classes for these entity classes tend to be the easiest to identify and because "migration" of key classes is initiated at these points in the model.

Key Class migration lays the foundation for beginning the conversion of the information model into a basic structural form. There are four basic rules which must be observed in Phase Three, all of which contribute to this transformation:

1. The use of "non-specific" relation class syntax is prohibited.

2. Key Class migration from independent to dependent entity classes is mandatory.

3. The use of attribute classes to represent attributes which could be null (or have no value) in an entity is prohibited.

4. The use of attribute class to represent attributes which might have repeating values (multiple values, or more than one value) at a time for a given entity is prohibited.

We have already discussed the first two rules in previous sections, so we will turn our attention to the last two at this point.

Figure 6-49 shows a refinement alternative diagram dealing with the application of the “no repeat” rule. Notice that the subject of the diagram reflects both the purchase order number and purchase order item numbers as being members of the Key Class of purchase order. However, evaluation of the way purchase order item number is used will show that a single purchase order (entity) can be associated or related to some number of purchase order items (entities). To properly depict this in the information model, a new entity class called “Purchase Order Item” would have to be created. To the relation class label, syntax and definition must be added. At this time, the true characteristics of the relationship between purchase orders and purchase order items begins to emerge.

Figure 6-50 shows a refinement alternative diagram dealing with the application of the “no null” rule. Part number has been inherited by purchase order item. This relationship was established because purchase order items are in some way associated with the parts, but the diagram asserts that every purchase order item is associated with exactly one part number. Investigation (or perhaps reviewer comment) reveals that not all purchase order items are associated with parts. In fact, some may be associated with services or other commodities which have no part numbers. This prohibits the migration of part number directly to the entity class purchase order item and requires the establishment of a new entity class in our example called “ordered part.”

Once a new entity class is established, Key Class migration must occur, as mandated by the migration rule and the modeler will once again “validate”the entity class/relation structure with the application of the “no null” and “no repeat ” attribute rules.

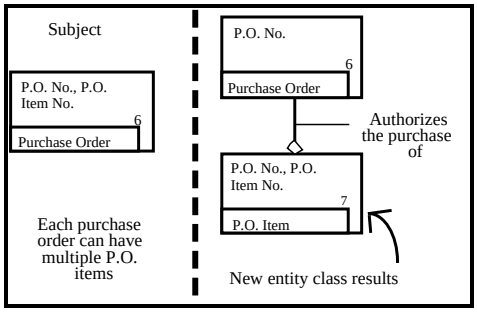


Figure 6-49. Phase Three - Applying the “No Repeat” Rule

6.5.5 Entity Class/Attribute Class Matrix

As Key Class members are identified, entries are made into the attribute class pool and into what is called the entity class/attribute class matrix. This matrix identifies the distribution and utilization of attribute classes throughout the model. Basically, it has the following characteristics:

1. All entity class labels are depicted on the side.
2. All attribute class labels are depicted at the top.
3. The use of attribute classes by entity classes is depicted in the adjoining vectors, as appropriate, using "keys" such as the following:

"O" = Owner

"K" = Key Class Member

"I" = Inherited

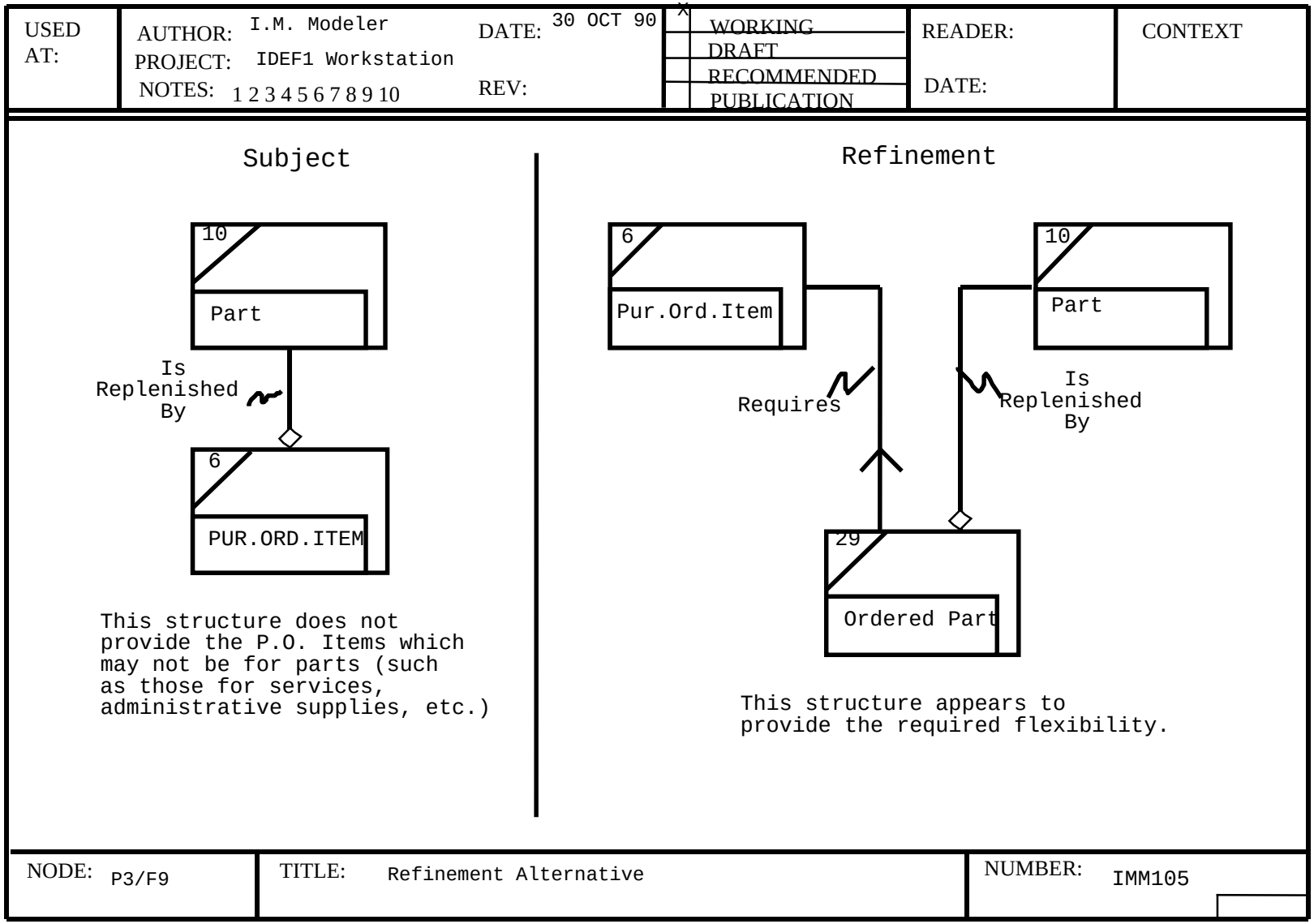


Figure 6-50. Refinement Alternative Diagram

A sample of an entity class/attribute class matrix is shown in Figure 6-51. This matrix is a principal tool in both maintaining model continuity and preparing model indices for inclusion in Phase Three and Phase Four documentation.

6.5.6 Key Attribute Class Definition

Once the key classes have been identified for the model, it is time for us to define the attribute classes which have been used as members of one or more key classes. In Phase Three, definitions are developed for these "key" attribute classes only. The same basic guidelines for these definitions apply as applied to relation class definitions and entity class definitions. They must be precise, specific, complete, and universally understandable.

Attribute class definitions are always associated with the entity class that "owns" the attribute class. That is, they are always members of the "owner" entity class set. Therefore, it is simply a matter of identifying those attribute classes for each entity class which have been identified as "owned" by the entity class and used in its (the owner's) Key Class. In Figure 6-51, those attribute classes are coded "OK" on the entity class/attribute class matrix.

There are two types of attribute class definition pages. One, shown in Figure 6-52, provides for the identification of the key class(es) for the entity class above the attribute class section. The other, shown in Figure 6-53, makes no provision for the entry of key classes. It is used principally as a continuation page for the attribute class definitions.

USED AT:

AUTHOR: I.M. Modeler DATE: 30 Oct 90

PROJECT: IDEF1 Workstation

NOTES: 1 2 3 4 5 6 7 8 9 10

WORKING / RECOMMENDED FOR PUBLICATION

READER DATE

CONTEXT

Entity Class	#	Attribute Class 1 PUR. Req. No.	2 Buyer Code	3 Vendor No.	4 Order Code	5 Change No.	6 Ship To Loc.	7 Vendor Name	8 Vendor Address	9 Contact Code	10 Item Name	11 Requester Code	12 Requester Name	13 Dept. Code	14 Ship Via	15 Buyer Name	16 Pur. Ord. No.	17 Purchase Req. Issue Date	18 Q.C. Attr. Code	19 Taxable Code	20 Resale Code	21 Pattern No.	22 Payment Terms
Pur.Requisition	1	OK	I									I											
Buyer	2		OK																				
Vendor	3			OK																			
Pur Order	4		I	I												OK							
Requester	6											OK											
Part	9																						
Pur. Req. Item	10	IK																					
Pur. Req. Line	12	IK																					
Approver	21																						
Part Source	22			IK																			

NODE: P3/T1

TITLE: Entity Class/Attribute Class Matrix

NUMBER: IMM76

Figure 6-51. Entity Class/Attribute Class Matrix

USED AT:	AUTHOR: I.M. Modeler PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 30 OCT 90 REV:	WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

KEY CLASS(ES):

ATTRIBUTE CLASS NAME	ATTRIBUTE CLASS LABEL	ATTRIBUTE CLASS I.D. NO.	ATTRIBUTE CLASS DEFINITION	ATTRIBUTE CLASS SYNONYM(S)

NODE: P1/E1 (G1)	TITLE: Attribute Class Definition(s)	NUMBER: IMM12

Figure 6-52. Attribute Class Defintion Page

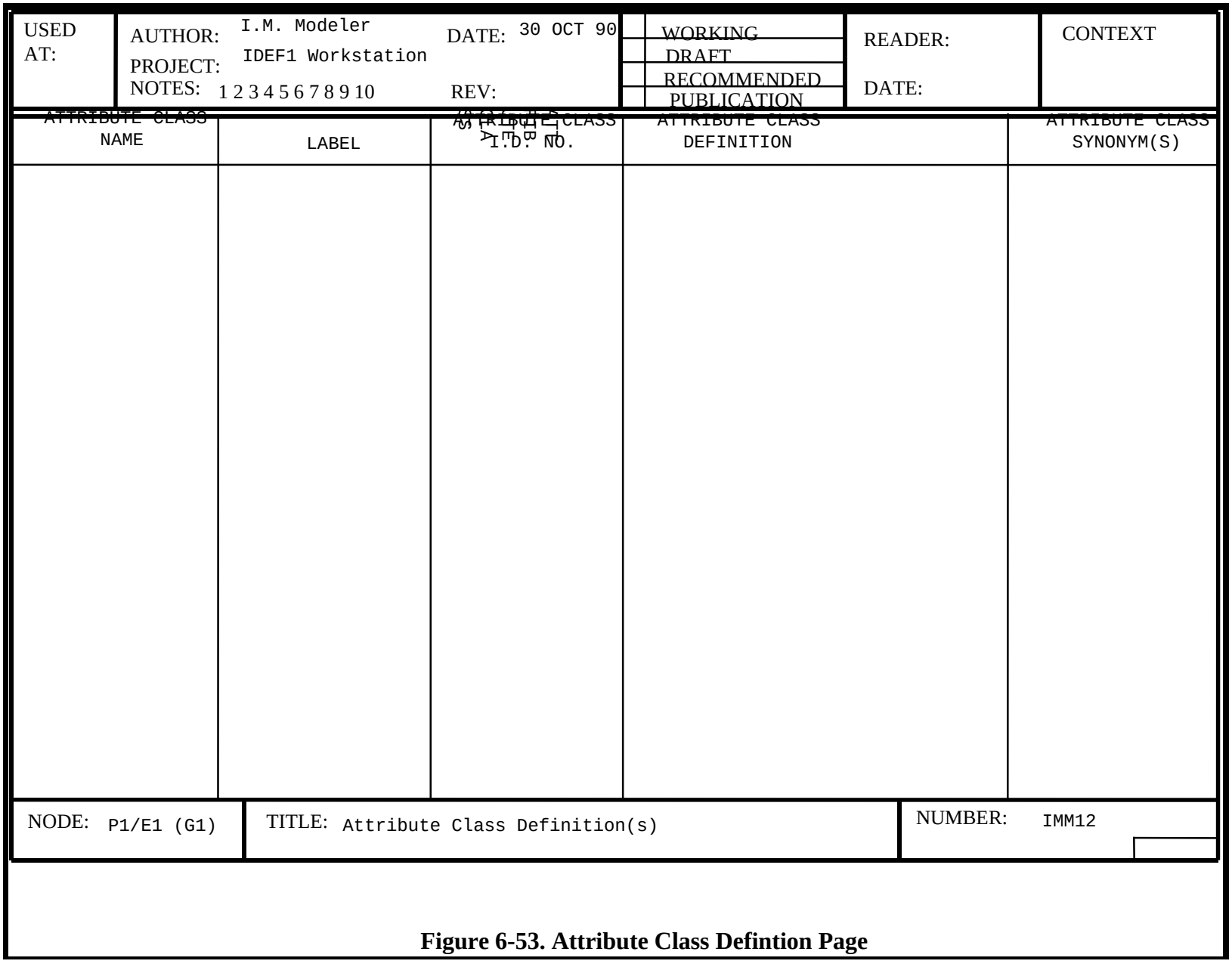


Figure 6-53. Attribute Class Defintion Page

6.5.7 Phase Three Formalization

Formalization of the Phase Three entity class sets can be a fairly extensive process. This is because there tends to be a number of new entity classes identified in Phase Three and there are several indices and cross-references to be completed, in addition. Basically, formalization of the Phase Three entity class sets involves the following steps:

1. Finalization of new entity class definitions and updating of the entity class pool (from Phase One).

2. Finalization of all attribute class diagrams, including the updating of the Relation Matrix, relation class definitions, and node cross-reference pages (from Phase Two).

3. Finalization of the attribute class definitions.

4. The development of inherited attribute class cross-reference pages.

5. The development of the attribute class migration indices.

6. Finalization of reference diagrams, including refinement alternatives and function views.

Attribute class diagrams are similar to entity class diagrams in some respects; one is that they reflect only those entity classes with which the subject entity class shares some direct relationship.

The characteristics of the attribute class diagram are:

1. Each attribute class diagram deals with only one subject entity class.

2. Each attribute class diagram reflects, in addition to the subject entity class, all entity classes related directly to the subject entity class.

3. "Independent" entity classes are at the top of the page and dependent entity classes are at the bottom of the page.

4. Only "specific" relation class syntax is used.

5. Each subject entity class box (one per attribute class diagram) reflects the Key Class identification of the entity class, with all Key Class members underscored, multiple attribute classes separated by commas and alternate Key Classes parenthetically enclosed.

6. Each subject entity class box (one per attribute class diagram) reflects all attribute classes which are inherited by the entity class, those which are not members of its Key Class being recorded below, and indented to the right of, the key classes.

This format is illustrated in Figure 6-54. The key class(es) is always underscored at the top of the entity class box (and left-justified). Any inherited “non-key” attribute classes are listed below the “key” classes and are indented noticeably from the left margin of the entity class(es) box. Non-key attribute classes are never underscored.

An example of the basic form used for attribute class diagrams is shown in Figure 6-55. Note that the subject entity class box is noticeably enlarged as compared to the subject entity class box used on entity class diagrams. This is to facilitate the additional information which must be enclosed in the entity class box on attribute class diagrams.

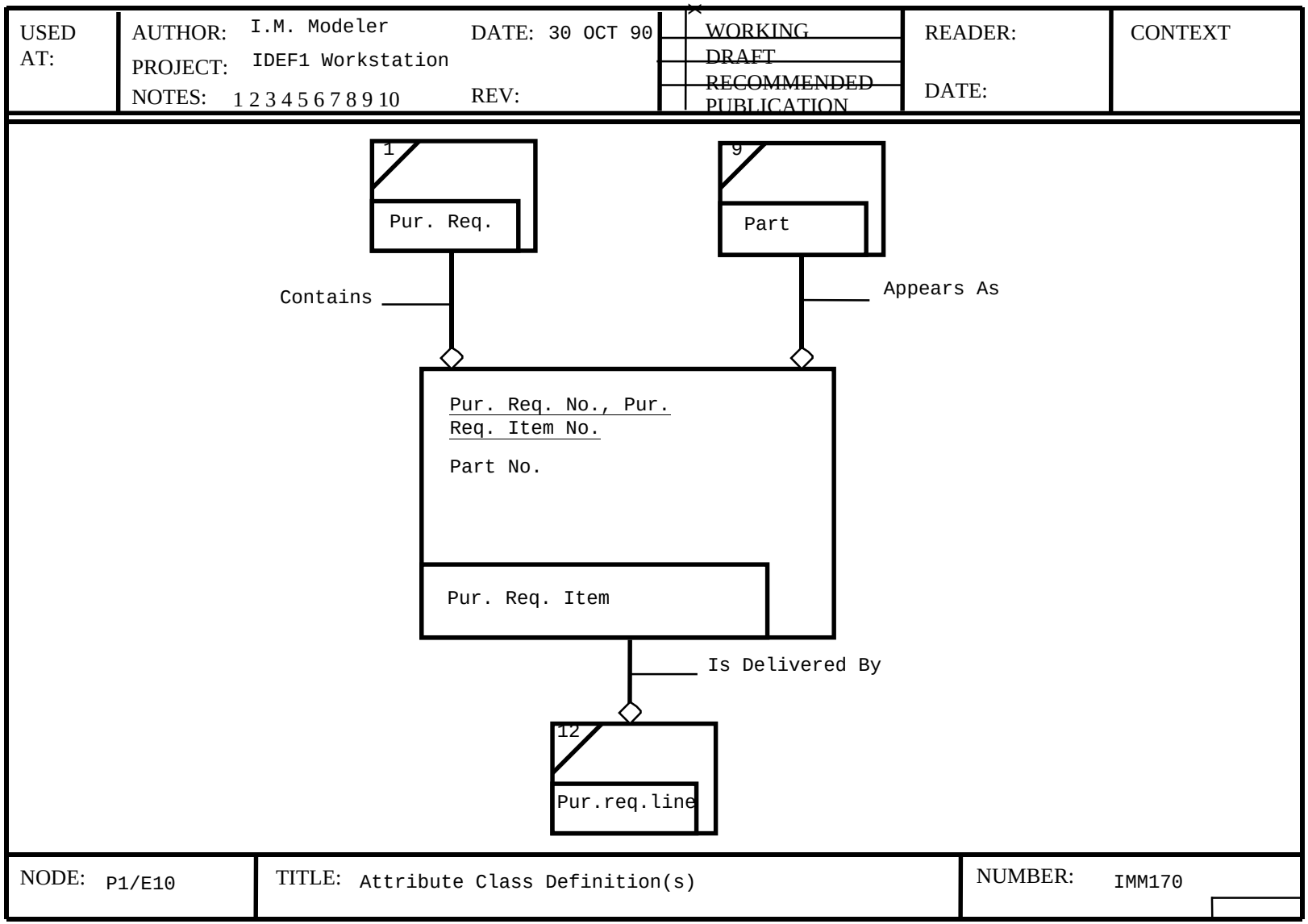


Figure 6-54. Attribute Class Diagram

USED AT:	AUTHOR: I.M. Modeler PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 30 OCT 90 REV:	X WORKING DRAFT RECOMMENDED PUBLICATION	READER: DATE:	CONTEXT

NODE: P1/E	TITLE: Attribute Class Diagram:	NUMBER:

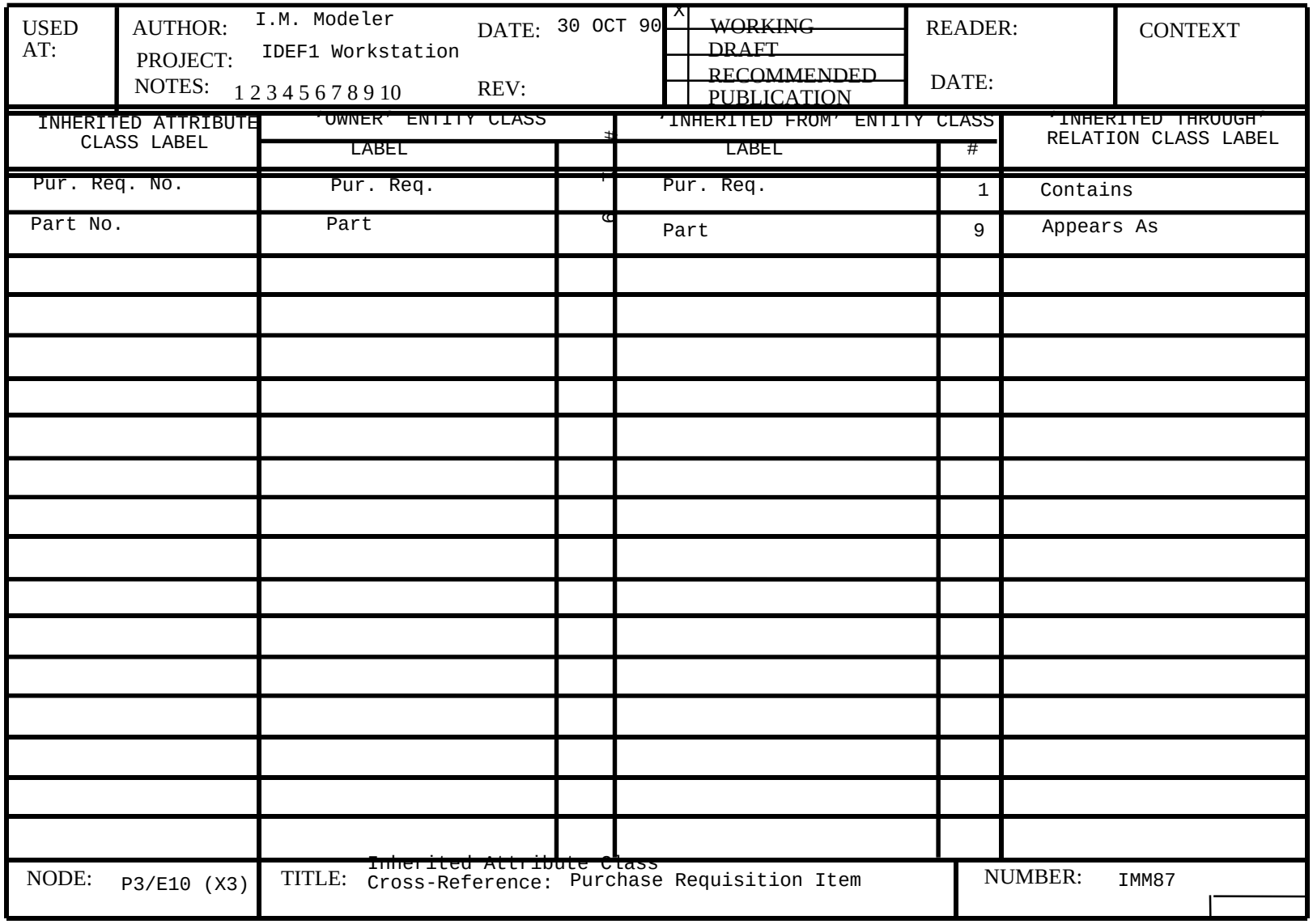
Figure 6-55. Attribute Class Diagram Form

USED AT:	AUTHOR: I.M. Modeler PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 30 OCT 90 REV:	~~WORKING~~ ~~DRAFT~~ ~~RECOMMENDED~~ ~~PUBLICATION~~	READER: DATE:	CONTEXT

OWNED ATTRIBUTE CLASS LABEL	'MIGRATED TO' ENTITY CLASS LABEL	#	OWNED ATTRIBUTE CLASS LABEL	'MIGRATED TO' ENTITY CLASS LABEL	#
Pur.Req. Item No.	Pur. Req. Line	12			

NODE: P1/E10 (G1)	TITLE: Attribute Class Migration Index: Purchase Requisition Item	NUMBER: IMM183

Figure 6-56. Attribute Class Migration Index



USED AT:

AUTHOR: I.M. Modeler
DATE: 30 OCT 90
PROJECT: IDEF1 Workstation
NOTES: 1 2 3 4 5 6 7 8 9 10
REV:

[X] WORKING DRAFT
[ ] RECOMMENDED PUBLICATION

READER:
DATE:

CONTEXT

INHERITED ATTRIBUTE CLASS LABEL	'OWNER' ENTITY CLASS LABEL	#	'INHERITED FROM' ENTITY CLASS LABEL	#	'INHERITED THROUGH' RELATION CLASS LABEL
Pur. Req. No.	Pur. Req.	1	Pur. Req.	1	Contains
Part No.	Part	9	Part	9	Appears As

NODE: P3/E10 (X3)
TITLE: Inherited Attribute Class Cross-Reference: Purchase Requisition Item
NUMBER: IMM87

Figure 6-57. Attribute Class Cross-Reference

The next task facing the modeler is the creation of the attribute class migration index and

the inherited attribute class cross-reference. Both of these documents are constructed based upon the entity class/ attribute class matrix.

The attribute class migration index reflects, from the "owner" entity class perspective, all of the entity classes wherein the various members of its Key Class (its "key" attribute classes) are used within the model. A sample migration index is reflected in Figure 6-56. To determine its content, the modeler first records an attribute class which is "owned" by an entity class and is a member of its Key Class. Then the modeler records the required information about each of the other entity classes with which the attribute class is shared, locating these by searching the vertical column of the matrix for the appropriate indicator. This process is repeated for all members of the "owner" Key Class.

The inherited attribute class cross-reference is from the perspective of the entity class which inherits in attribute class. It reflects how the attribute class arrived at the subject entity class. A sample of an inherited attribute class cross-reference is shown in Figure 6-57. To determine its content, the modeler first records all attribute classes which are inherited by the entity class and then, using the entity class/attribute class matrix, the "owner" of each one is identified and recorded. Using the attribute class diagram, the modeler records the information about the entity class and relation class through which the attribute class was directly inherited.

6.5.8 Phase Three Kits

Figure 6-58 illustrates the general structure of a typical Phase Three kit. Note that the structure of the review kit has become progressively more complex since Phase One.

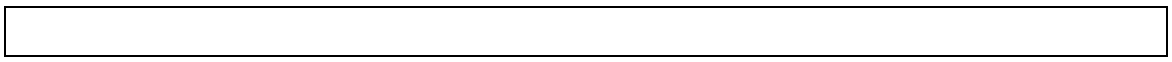
The basic structure of the Phase Three kit, though more complex than Phase Two, is similar in several respects. For example, the cover sheet, kit overview, node cross-reference and reference only entity class definitions are handled in the same way. The primary difference is the injection of the function view (which is optional) and the additional material within the context of each entity class set. In total, the typical Phase Three kit will run in the vicinity of 40 to 44 pages. This allows some six to ten pages available to the modeler for the insertion of general text or reference diagrams pertaining to subjects other than the entity class sets within a kit.

In Phase Three, there are two types of kit overviews. If the kit is around a function view, then the same guidelines as used in Phase Two apply to structuring of the kit overview. If a function view is to be used, then the kit overview must be a subset of the function view employed.

Phase Three kits, like all other IDEF modeling kits, are prefaced with a kit cover sheet. A sample Phase Three kit cover sheet is shown in Figure 6-59.

When a Phase Three kit is organized around a function view, the subject kit overview diagram, is simply a subset of the function view diagram representing or reflecting those entity classes which are subjects of the kit. The function view diagram is included to provide the context within which the information contained in each entity class set is to be validated. The focal point of the review is intended to be validation of the relation classes reflected in the context of the function view and validation of all attribute class information within each entity class set. The secondary issue is the validation of all other relation classes shared by each subject entity class in the kit. This is because the remaining relation classes should be topics of other function views and submitted for review and validation in other kits.

A sample kit overview, representing a subset of a function view around which the kit was organized, is shown in Figure 6-60. Figure 6-61 is the function view it is based on. There has been an attempt to use the same lines and spacing on the kit overview as used in the function view; this is to achieve a degree of highlighting effect and to focus attention on the subject of the kit within the context of the function view used.



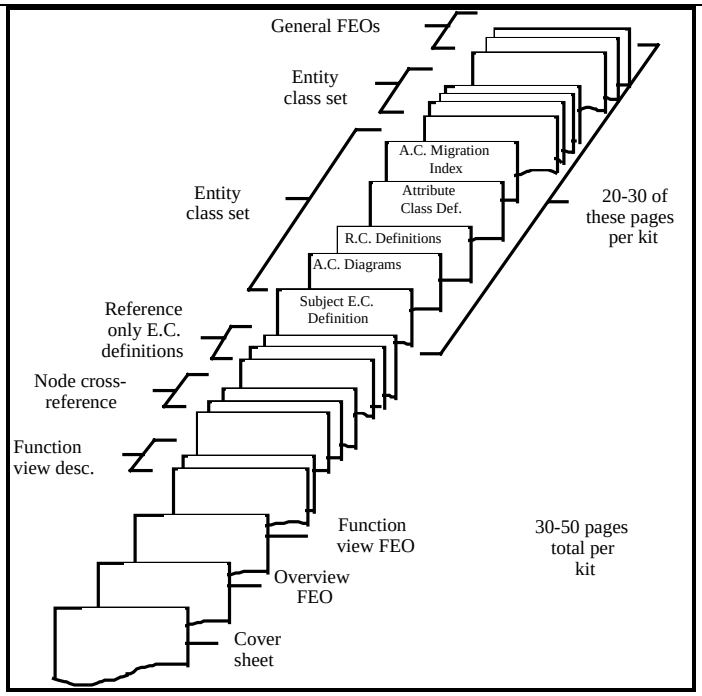


Figure 6-58. Phase Three - Kit Structure

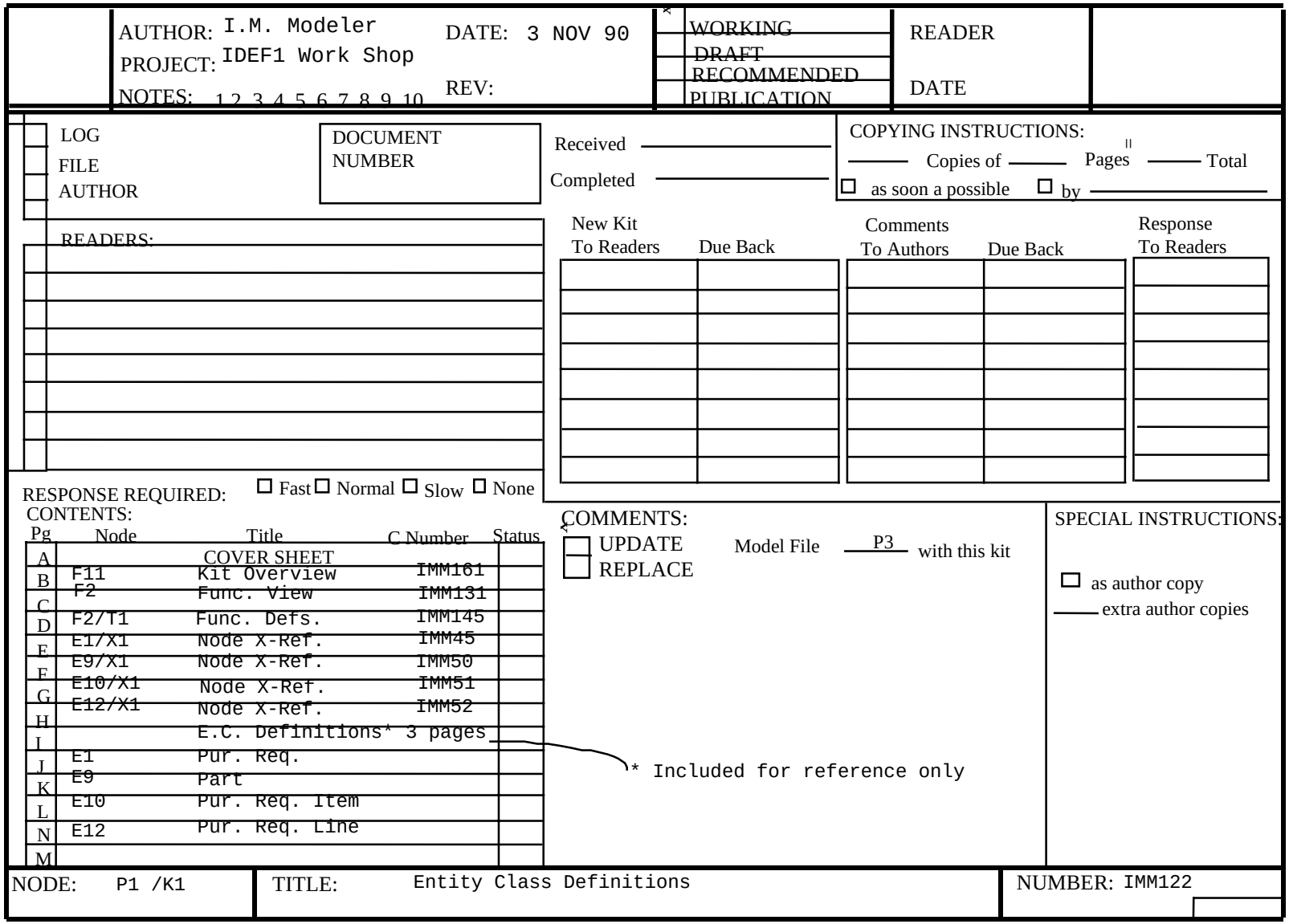
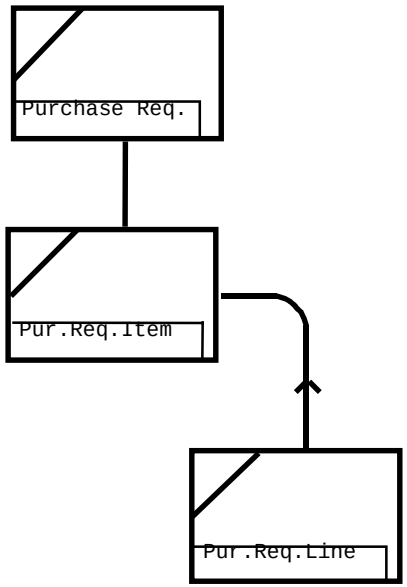


Figure 6-59. Phase Three Kit Cover Sheet

USED AT:	AUTHOR: I.M. Modeler PROJECT: IDEF1 Workstation NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 30 OCT 90 REV:	~~WORKING~~ ~~DRAFT~~ ~~RECOMMENDED PUBLICATION~~	READER: DATE:	CONTEXT



NODE: P2/E1	TITLE: Entity Class Diagram: Purchase Requisition	NUMBER: IMM23

Figure 6-60. Kit Overview

As in prior phases, there are some specific kit sizing parameters applicable to Phase Three. Phase Three kits:

1. Should contain 30 to 50 pages total.
2. Will typically contain 2 to 4 entity class sets.
3. Should require no more than 1 1/2-2 hours of review time per reader.

Note that the total number of pages in an entity class set has increased markedly from Phase Two. The typical entity class set in Phase Three will contain 10 pages.

The addition of new entity classes particularly in such volume as expected in Phase Three, makes it advisable for the modeler to mark each page that is for reference only and each entity class definition that is released for the first time.

Last, is the importance of handling new entity classes which are developed in Phase Three. The basic rule to be employed is that any new entity classes developed in Phase Three must meet all of the Phase One and Phase Two requirements before they can be incorporated into the model. This includes entry into the entity class pool, development of entity class definitions, inclusion in the Relation Matrix, development of relation class definitions, etc.

6.6 Phase Four – Attribute Class Population

The last phase of the basic information modeling process is Phase Four. The objective of this phase is to determine the "owner" entity class of each "non-key" attribute class represented in the attribute class pool. Although, on the surface this last phase appears to be relatively simple, it can result in some rather appreciable changes in the model structure.

Perhaps the most important rules to remember during Phase Four are:

1. An attribute class has only one "owner" in the model.
2. Non-key attribute classes cannot migrate.
3. The "no null" attribute rule.
4. The "no repeat" attribute rule.

The Phase Three construction of the attribute class pool was initiated and an entity class/attribute class matrix was built. The matrix is only partially completed at this point in

time, indicating only those attribute classes which were used as members of one or more key classes in Phase Three. Attribute classes which were not used as members of any Key Class in Phase Three are the targets of Phase Four.

Phase Four concentrates on the further delineation of already established materials, rather than producing an appreciable quantity of new material. There are new attribute class definitions which are added and some number of refinement alternative diagrams or function views. It should also be expected that some quantity of new entity classes will emerge during Phase Four, primarily derived as a result of the application of the refinement rules already exercised in Phase Three, reapplied as the rest of the attribute classes are distributed through the model.

A partial list of the products resulting from Phase Four is reflected below:

1. Expanded Attribute Class Pool
2. Revised Attribute Class Definitions
3. Revised Entity Class/Attribute Class Matrix
4. Refined Attribute Class Diagrams
5. Revised Inherited Attribute Class Cross-Reference
6. Revised Attribute Class Migration Index
7. Refinement Alternative Diagrams
8. Revised Function View Diagrams
9. Phase Four Kits

The basic processes employed in Phase Four are as follows:

1. Identify “non-key” attribute classes from Phase Three.
2. Identify the “owners” of the non-key attribute classes.
3. Define the non-key attribute classes.
4. Refine the relation classes.
5. Formalize the Phase Four entity class sets.
6. Revise the function views (FEOs).

7. Build and distribute Phase Four kits.

6.6.1 Phase Four Process

The first step in the construction of Phase Four material is the identification of non-key attribute classes from Phase Three. This is relatively straightforward. Any attribute class–with no usage reflected on the entity class/attribute class matrix from Phase Three–is in fact a “non-key” attribute class at this point.

The next step requires that each of these non-key attribute classes be assigned to one “owner” entity class. The “owner” entity class for many of them will be obvious. For example, in the case of vendor name, the modeler should be able to readily associate this attribute class with the entity class vendor, but there may be some attribute classes which will cause the modeler difficulty in locating their “owner” entity classes.

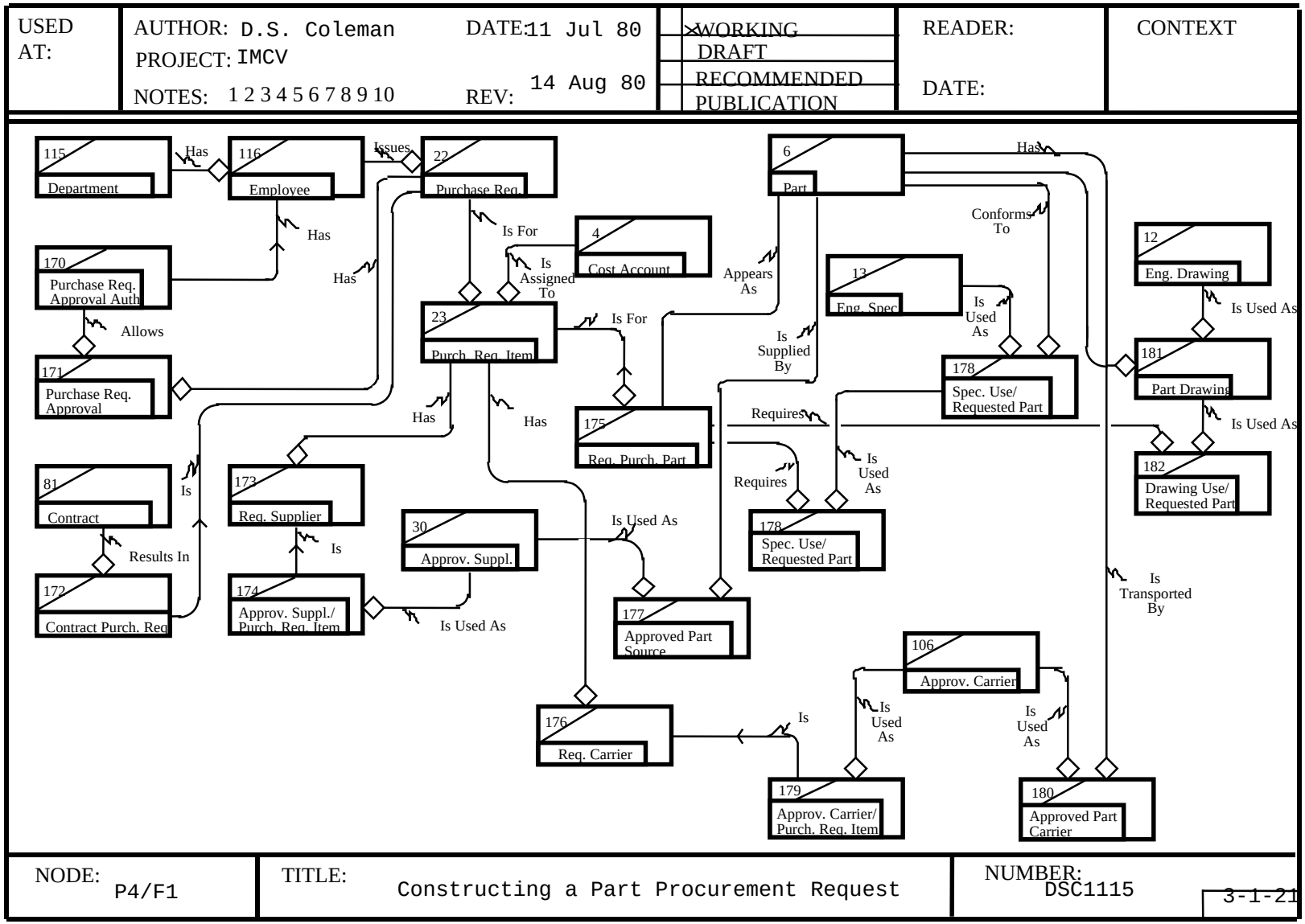


Figure 6-61. Function View

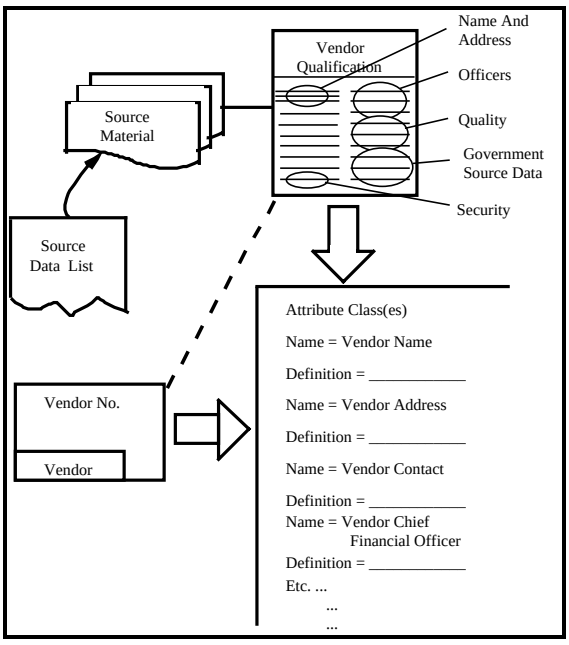


Figure 6-62. Non-Key Attribute Class Population

If the modeler is not certain of the "owner" entity class of an attribute class, it may be advisable to refer to the source material form which the attribute was extracted. This will aid in the determination of the "owner." In Phase Zero, the source data list was established and became the foundation for the attribute class pool. The source data list, then points the modeler to the location(s) where the attribute represented is used in the original source material. This is illustrated in Figure 6-62. By analyzing the usage of the attribute class in the source material, the modeler will be able to more easily determine the "owner" entity class in the information model. The modeler should keep in mind that the governing factor for determining "ownership" of the attribute classes is reflected in the source material. As each attribute class is assigned to its owner entity class, the attribute class/entity class matrix must be updated accordingly. This matrix, like the Relation Matrix, is one of the primary working tools of the modeler for managing the continuity of the information model.

For each of the attribute classes identified in Phase Four, a definition must be developed. The principles governing other definitions used in the information model and particularly those in Phase Three, apply here as well. The definitions must be precise, specific, complete, and universally understandable. These attribute class definitions are produced in the same

format as the attribute class definitions from Phase Three. They are simply added to the attribute class definitions in the owner entity class set.

The modeler is now ready to commence with the Phase Four refinement of relation classes. The same basic rules apply to this refinement as applied in Phase Three. The application as the “no null” and “no repeat” attribute rules introduced in Phase Three are now applied. As a result, the modeler can expect that some number of new entity classes will be found. As these entity classes are identified, the Key Class migration rule must be applied, just as it was in Phase Three.

The only difference in applying the no null and no repeat attribute class rules in Phase Four is that they are applied primarily to the “non-key” attribute classes as well. Figure 6-63 illustrates the application of the “no null” attribute rule to a “non-key” attribute class. Figure 6-64 illustrates the application of the “no repeat” attribute rule applied to non-key attribute classes.

As new entity classes emerge, they must be entered in the entity class pool, be defined, reflected in the Relation Matrix, etc. They must meet all of the documentation requirement as earlier phases in order to qualify for inclusion in Phase Four material.

With the Phase Four material gathered and prepared, the task of formalizing the Phase Four entity class sets can commence. This process is almost identical to the corresponding process of Phase Three. This is primarily because the basic changes to the model during Phase Four are:

1. Completion of attribute class identification.
2. Completion of the attribute class distribution.
3. The completion of definitions for all attribute classes.
4. Structural changes resulting from the application of the refinement rules to the attribute classes.

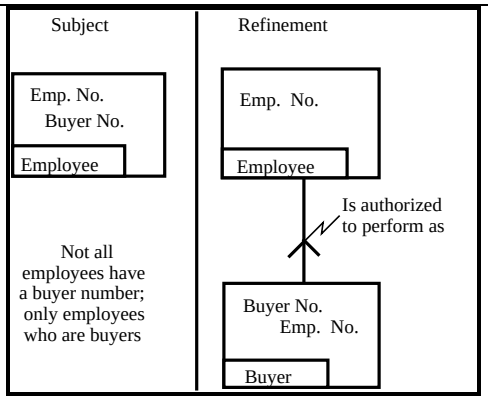


Figure 6-63. Phase Four

Primarily, formalization of the entity class sets involves the revision and update of material which has been previously constructed. The steps involved in the Phase Four formalization of those previously defined materials are:

1. Formalize new entity class definition
   - A. Update entity class pool
2. Finalize attribute class diagrams
   - A. Update relation class matrix
   - B. Update relation class definitions
   - C. Update node cross-reference
3. Finalize attribute class definitions
4. Finalize inherited attribute class cross-reference
5. Finalize attribute class migration index
6. Finalize reference diagrams

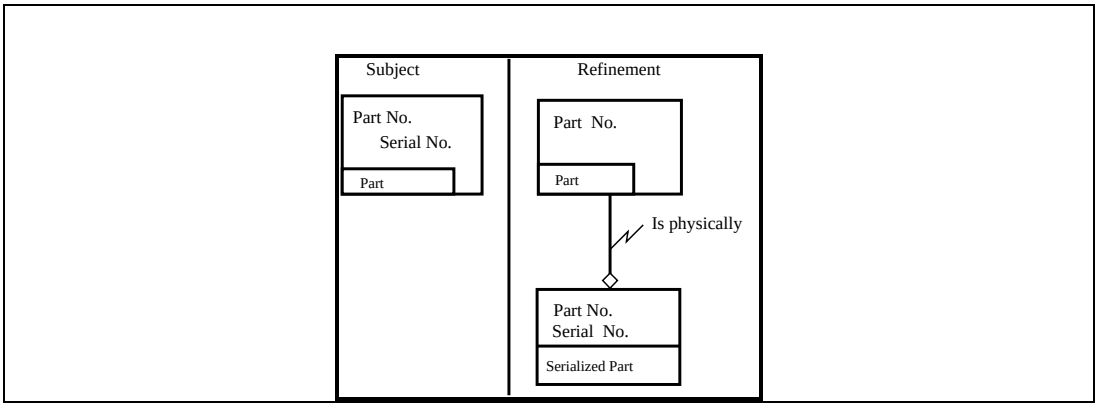


Figure 6-64. Phase Four - Applying The "No Repeat" Rule

If the modeler has developed function views in Phase Three, then the structural changes resulting from Phase Four would necessarily cause revision of some number of function views. In Phase Three it was emphasized that the function views reflect some portion of the existing model. If a change to the existing model effects entity class or relation class structures included within the scope of any function view, then the affected function view(s) must be revised. Correspondingly, the entity class/function view matrix developed in Phase Three must be revised to reflect these changes.

6.6.2 Phase Four Kits

Once the author has completed the formalization of the Phase Four entity class sets, the construction of the Phase Four kits can proceed. Phase Four kits are quite similar to Phase Three kits in their general construction. Once again, because of the size and complexity level of the model, the modeler may choose to release kits oriented around a function view. The basic parameters for successfully using the function view remain the same as they were in Phase Three. In addition, should the modeler choose to produce kits oriented around the function views, the construction of these kits is essentially identical.

The basic kit structure is illustrated in Figure 6-65. Also, a sample of a cover page is shown in Figure 6-66. The same basic guidelines for kit structuring applied in Phase Three can be applied to Phase Four.

Phase Four kits:

1. Should be 30 to 50 pages in length, total.
2. Will typically contain 2 to 4 entity class sets.
3. Should require no more than 1-1/2 to 2 hours of review time from each reader.

The volume of pages in each entity class set in Phase Four will not have varied significantly from Phase Three entity class sets, but the volume of entity class sets, in total, will have increased. The average entity class set in Phase Four will contain about 11 pages.

Note that the kit sizing and structuring characteristics in Phase Four are almost identical to those in Phase Three, but because of the slight growth in the number of pages per entity class set, the average Phase Four kit size will be between 45 and 50 pages, typically. This still allows some 2 to 3 pages for the modeler to use for general text and reference diagrams, if that is desirable. Again, the modeler should take into account the amount of time requires for and the difficulty in reviewing the material.

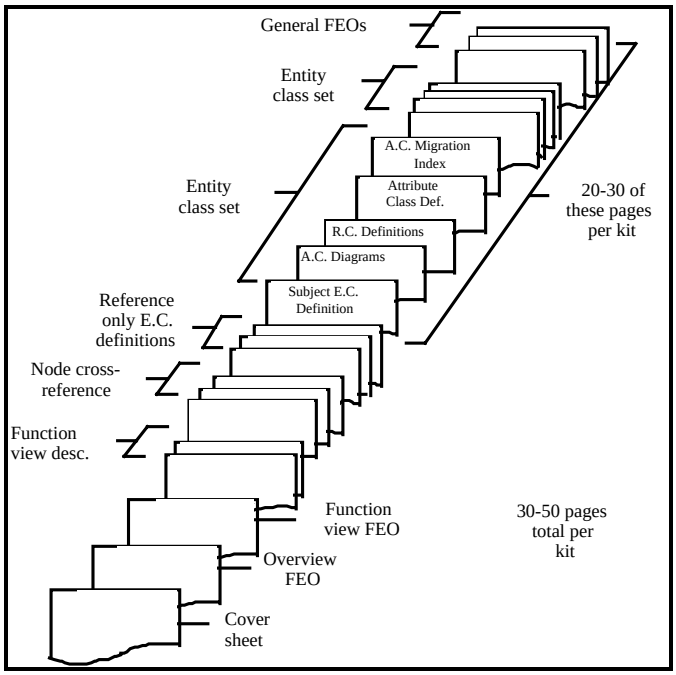


Figure 6-65. Phase Four - Kit Structure

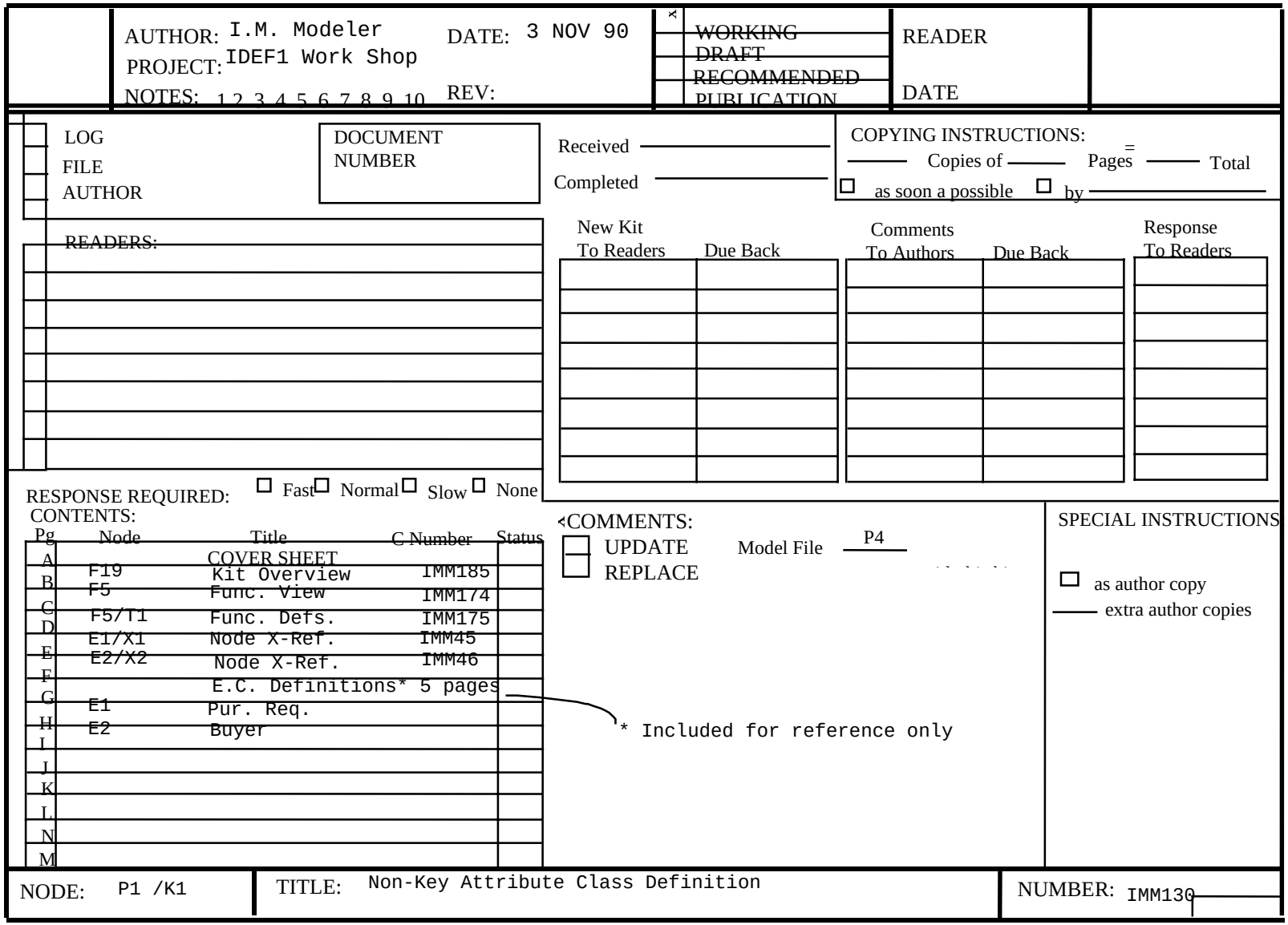


Figure 6-66. Phase Four Kit Cover Sheet

The same basic rules regarding the expansion of the model apply to Phase Four, as they have applied to prior phases. Any new entity classes discovered (derived) in Phase Four, if they are to remain in the model, must be registered in the entity class pool, be assigned their own unique number, be defined, etc. All of the rules and requirements of earlier phases apply to all new entity classes, relation classes, and attribute classes. These must be met before they can be included in Phase Four diagrams and kits.

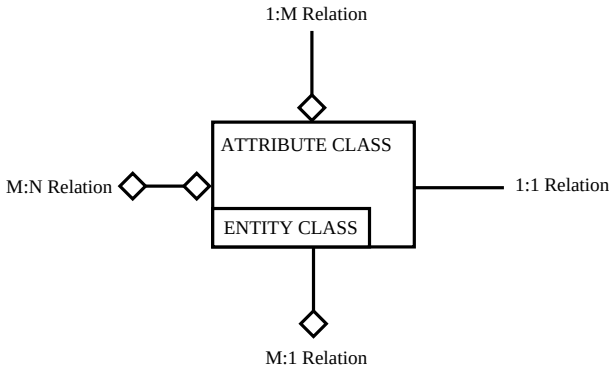
Once the modeler has begun the construction of Phase Four kits, they can be distributed to reviewers for comments. It is through this process of review, comment, and iteration that the resultant model is validated

6.6.3 Conclusion

Upon completion of Phase Four, the modeler has produced a basic information model. If all of the methodology rules have been applied correctly throughout the development, the model will be representative of the fundamental information structure of the enterprise modeled. Each entity class will represent a non-redundant collection of information; each entity class pair sharing a relation class will convey some non-redundant meaning in the model.

At this point, the information model is in a form which will facilitate basic translation into any data base management system currently on the market. This does not imply that the resultant information model, at the end of Phase four, is a database design. Rather, the model represents a stable information structure–a stable set of rules. Regardless of the technical specifics of any database which eventually results, its architecture must adhere to the rules of information structure and meaning reflected in the information model. It is through this medium that integration of information in the manufacturing enterprise can be approached.

Section 7.0 Data Collection For IDEF Modeling



7.0 Data Collection For IDEF Modeling

7.1 Introduction

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

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

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

Obtaining information from an expert has been formalized in an interview process. This provides steps to follow, so that an interview can be conducted without unduly influencing the expert with information already obtained by the interviewers.

A key part of interviewing is to record the information obtained. This can be done either as informal notes or as diagram sketches.

7.2 The Interview Process

The purpose of an interview is to gather information from an individual who possesses an expertise considered important to the analytical effort. There are four types of interviews

that might be conducted during the course of performing the analysis phase of an IDEF project.

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

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

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

4. IDEF Author/Commentor Talk Session. This type of interview is used to resolve problems which have surfaced during the construction of an IDEF model.

The reason for identifying types of interviews is that during the course of performing an actual interview, ingredients of each type appear. The respondent might tell the interviewer facts about a given system in terms of problems. Also, the respondent might identify problems in terms of solutions to the problems. The interviewer, by constantly classifying the respondents remarks, can obtain the maximum useful information from the interview.

7.3 The Interview Kit

It is recommended that a "standard" Interview Kit be used for recording the interview. It may be stored in an Interview File and it may be distributed to appropriate individuals. This distribution might include other members on the Analysis team or even the interview respondent for corrections, additions, and deletions. The interview kit would contain:

1. Cover Page (Kit cover)

2. Interview and Record Follow-up
   - A. Interviewer Name (IDEF Author Name)
   - B. Interview Date (IDEF Diagram Date)
   - C. Interview Duration (Start time, End time)
   - D. Respondent Name
   - E. Respondent Title and Organizational Responsibility
   - F. Respondent Telephone Number and Extension

G. Additional Sources of Information Identified

1. Documents – Title and Location

2. Other Interviewees – Name, Title, Organizational Responsibility, Address, Telephone number

H. Essential Elements of Information – a Summary of the key points covered in the interview.

I. Follow-up questions and/or areas of concern either not covered during the interview, or postponed

J. New Terms for Project Glossary

3. Entity and Attribute Pool Candidates

4. Interview Agenda (Developed in preparation of Interview – This is covered in the following section)

5. Interview Notes and Rough Diagrams

7.4 Interview Guidelines

There are five stages to the successful interview; each must be performed in order to assure that the most information is obtained and recorded in the least amount of time.

1. Preparation
2. Initialization
3. Interview
4. Termination
5. Finalization

In each stage of an interview there are certain basic activities which must be performed. Additionally, associated with each stage, there exist psychological aids which will help the interviewer establish an atmosphere of professionalism and trust with the respondent.

7.4.1 Interview Preparation

By thinking through certain key interview needs before the interview, a more organized and efficient dialogue can be achieved. Preparation for an interview should contain, but is not limited to, the following activities:

1. Select Interviewee

   A. From areas of responsibility

   B. From recommendations of others

   C. From various levels of the organizational hierarchy – upper levels useful for big picture, lower levels for detail information, and middle levels for bridging the gap

2. Make Appointment

   A. Short duration – 1/2 to 1 hour

   B. Not immediately before lunch, nor late afternoon

   C. Identify purpose of interview

   D. Explain interviewer role

3. Establish Tentative Agenda

   A. Topical areas – used as a foundation for interview (this helps prepare "broad general questions")

   B. Specific questions

4. Review applicable background information

5. Review appropriate terminology

6. Insure coordination with other interviews

   Check interview file to ascertain that the respondent has or has not been previously interviewed. If the interview is a follow-up interview, then examine the results of previous interviews.

7. Fill out Interview Record and Follow-up with pertinent information

8. Make out Interview Agenda

7.4.2 Interview Initialization

This stage of the interview is directed at establishing a rapport between the interviewer and respondent. The courtesy permitted by the respondent at the start of an interview is usually short. This time is important in motivating the respondent to help the interviewer. This stage of the interview should contain the following topics:

1. Provide respondent with a tangible means of introduction e.g., a business card (this removes doubt on the part of the respondent as to

how to pronounce or spell the interviewer's name and can therefore remove a frequent cause of respondent embarrassment).

2. Establish purpose of interview

   A. Expand on information provided in initial contact.

   B. Establish point of view for the interview. Use interview type 1, 2, 3 or 4 as a basis.

   C. Establish purpose of the interview–even if the interview is a follow-up interview.

3. Establish the acceptability of note taking. The respondent may require assurance of confidentiality.

4. Establish the Expert/Author relationship – alleviate the fear that the interview will be used to tell the respondent how to do his job, or that the respondent's job is in jeopardy.

5. Start with broad general questions which will get the respondent talking – these should be based upon the topical areas identified in the agenda.

6. Assess the respondent's ability to provide pertinent information – if the information is too general or too detailed for the stage of the IDEF model being prepared, evaluate respondent's ability to contribute. Terminate the interview if necessary–it may be a waste of both the interviewer's and respondent's time.

7. Begin to formulate specific questions which complement the agenda.

8. Write, don't talk.

7.4.3 Conducting the Interview

While it is not useful to define questions to ask during an interview, it is possible to identify guidelines that should be considered during the interview. The first set of guidelines deals with the qualification of the information being obtained. The second set of guidelines relate to the stimulation of information flow.

Information Qualification: The human mind can comprehend at double the rate at which people speak. The danger in interviewing is that this rate difference is typically used by the listener to think about what should be said in response instead of about what is being said.

To assist the interviewer in thinking about what is being said, there is a series of questions which may be used to help the interviewer keep his mind on the information being provided:

1. What supporting facts are being provided for the main points being discussed?

2. How recent is the information?

3. How complete is the information?

4. Do I really understand what is being said?

5. Is the level of detail being presented appropriate for my purpose?

6. Are there areas being omitted?

7. Has this information been discussed with someone else?

8. How important is this information?

9. Are side-topics being discussed?

10. Has the interview viewpoint changed?

: The following set of guidelines can be used to stimulate the respondent into providing maximum reliable information.

1. Keep extraneous comments and conversations to a minimum. The interview is used to obtain information, not make friends, or sell ideas.

2. Be aware of the respondent's failure to identify problem areas in the environment. This may indicate that the respondent is not at ease with the interviewer.

3. Provide the respondent time to think. Do not suggest answers or ask another question. A pause in the interview is useful to allow the respondent to recall vital pieces of information.

4. Avoid outside distractions which tend to "uncouple" the train of thought. If at all possible, conduct interviews outside of the respondent's normal habitat.

5. Be aware of internal distractions, signs that the respondent is not comfortable or at ease with the interview.

6. Try to determine if the information being obtained is fact or opinion.

7. Encourage elaboration by asking for a rephrasing or a summary of the information presented.

8. Ascertain the respondent's background and association with the subject matter being discussed. Valuable insight into the respondent's remarks can be obtained knowing his relationship to the organization and existing systems.

9. Do not enter into or encourage sarcasm and humor.

10. Do not mention or discuss any interview with another person.

11. Record all questions asked by the respondent. The interviewer should answer all questions except those dealing with user–organization management, plans, or personalities.

12. Show interest in what the respondent is saying.

13. Concentrate on the unfamiliar and difficult aspects of the subject being discussed. Avoid the obvious.

14. Be alert for the inconsistent or incorrect use of words. Ask for definitions for any unfamiliar or questionable term. Record the definition for the project glossary.

15. Do not contradict the respondent even if facts do not support what is being said. Use the Kit Cycle to resolve such conflicts.

16. Be humble. The respondent is the expert, not the interviewer.

17. Postpone subjects which cannot be fully covered within the agreed upon time frame. Do not extend the interview time, but rather make another appointment.

18. Appreciate different opinions on the same subject. Use IDEF to show these opinions and to resolve conflicts.

19. Stimulate the respondent with pertinent open ended questions.

7.4.4

The interview should be terminated because of the four following reasons:

1. The information being obtained in the interview is not appropriate.

2. The time limit has been reached.

3. The interviewer has been saturated with information.

4. There is a clash of personalities between the interviewer and the respondent.

Depending upon the cause of termination, the following topics should be considered during the termination of the interview:

1. The interview should not be closed abruptly, but rather should end with a few minutes of informal discussion.

2. The main points of the interview should be summarized.

3. Areas of concern which have been postponed or not covered should be identified.

4. A follow-up interview, if necessary, should be arranged.

5. The respondent should be asked to recommend other persons who should be interviewed.

6. If the interview notes are to be reviewed by the respondent prior to distribution, this fact should be mentioned during the termination.

7. The respondent should be thanked for his time and effort.

7.4.5 Finalization

This stage of the interview is directed at assuring that the information obtained during the interview is properly recorded and disseminated to the project team. The vehicle used to accomplish the finalization of an interview is the Interview Kit. If note taking was not permitted by the respondent, the interviewer should, upon termination of the interview, immediately write down the salient points discussed. Finalization of the interview includes the following:

1. Identify additional sources of information.

2. Summarize the Essential Elements of Information.

3. Identify new terms for the project glossary.

4. List the follow-up questions and areas of concern either postponed or not covered during the interview.

5. Complete Entity and Attribute Pools.

6. Expand on any notes with any information recalled during the review.

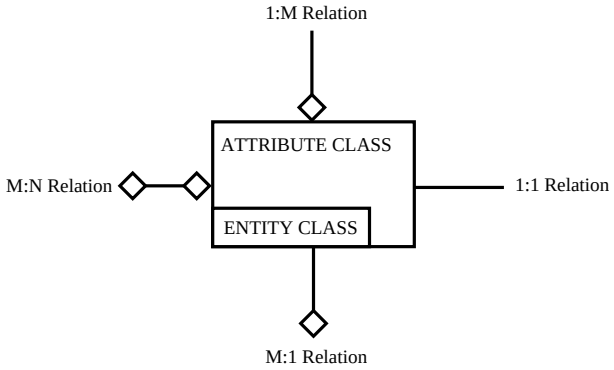
7. Prepare rough IDEF diagrams that reflect the information obtained.

8. Identify in the Interview Kit any assumptions being made or any items which are not clear.

9. Publish and distribute the Interview Kit.

10. Add persons name, area of expertise, phone number, and address to the expert and commentor list which were mentioned in the interview.

Section 8.0 IDEF1 Glossary



IDEF1 Glossary

Acceptance Review Committee – One of the members of the functional organization whose responsibility is to provide guidance and arbitration over the modeling effort and to pass final judgement over the completed product (i.e., model acceptance).

Alternate Key Class – An alternate Key Class is a Key Class which is exactly equivalent to another Key Class and can be used as a unique identifier of an entity class with precisely the same effect as the other Key Class.

Attribute – A property (characteristic) of an entity; an attribute is composed of a name and value and is one element of information known about an entity.

Attribute Class – A collection of attributes of the same name which apply to all entities of the same entity class; attribute class names are singular, descriptive nouns.

Attribute Class Diagram – A diagram containing one "subject" entity class and all entity classes directly related to it, with the key attribute class(es) and non-key attribute class(es) of the subject entity class displayed.

Attribute Class Population – That effort by which the "ownership" of attribute classes is determined.

Author Conventions – The special practices and standards developed by the modeler to enhance the presentation or utilization of the model. Author conventions are not allowed to violate any methodology rules and do not represent "official" standards of practice.

Data Collection Plan – The plan which identifies the targets–the functions, the departments, the personnel which are the sources of the material used for the development of the model.

Entity – An object, either physical or conceptual, which exists within the scope or the model, is uniquely identifiable and has one or more attributes which define its specific characteristics.

Entity (IDEF1) – A collection of information about a specific object (entity).

Entity Class – A collection of entities (IDEF1) which are described using the same kind of information.

Entity Class Diagram – A diagram which depicts a "subject" entity class and all entity classes directly related to the subject entity class.

Entity Class Set – A collection of information about an entity class, representing all of the information that is known about the entity class.

Expert Reviewer (Commentor) – One of the members of the functional organization whose expertise is focused on some particular activity within the manufacturing enterprise and whose responsibility it is to provide critical comments on the evolving model.

FEO – An acronym meaning For Exhibition Only; it is one vehicle by which supportive or explanatory information is provided for the model, via some combination of drawings, text, etc.

IDEF1 – An acronym standing for ICAM DEFinition Method (Information Model Methodology): this method is used for the expression of the structural characteristics of information

IDEF1 Kit Cycle – The regular interchange of portions or the model in development between the modeler and the readers/expert reviewers, the purpose of which is the isolation and detection of errors, omissions, and misrepresentations.

IDEF1 Model – A representation of the structural characteristics of information; a requirements statement which reflects the basic nature of information.

Key Class – One or more attribute classes which are used to uniquely identify each member of an entity class.

Key Class Migration Rule – One of the rules of the IDEF1 Method which defines (migration) of key classes between a related "pair" of entity classes.

Modeler (Author) – One of the members of the Functional Organization whose responsibilities include the data collection, education and training, model recording, and model control during the development of the model; the modeler is the expert on the IDEF modeling methodology.

"No Null" Attribute Rule – One of the rules of the IDEF1 methodology which establishes specific action requirements which must be met whenever a situation arises in which an attribute class is used to represent an attribute value which would not be available for any member of the entity class at any time.

"No Repeat" Attribute Rule – One of the rules of the IDEF1 methodology which establishes specific action requirements must be met whenever a situation arises in which an attribute class represents more than one attribute value for any member of the entity class at a time.

Phase Zero – The initial efforts of the modeling activity in which the Context Definition is established–project definition, data collection plan, author convention standards, etc.

Phase One – The second in the orderly progression of modeling efforts during which the entity classes are identified and defined.

Phase Two – The third in the set of orderly steps of the modeling efforts, during which the relation classes are identified and defined.

Phase Three – The fourth step in the orderly progress of model development, during which the key classes are identified and defined.

Phase Four – The fifth effort in the progression of orderly model development, during which the "non-key" attribute classes are identified and defined.

Project Manager – One of the members of the Functional Organization whose responsibilities include administrative control over the modeling effort–the duties include: staffing the Functional Organization, setting the scope and objectives, chairing the acceptance review committee, etc.

Relation – A meaningful association between two entity classes.

Relation Class – The manner in which the members of one entity class are associated with (or related to) members of another entity class (or members of their own entity class).

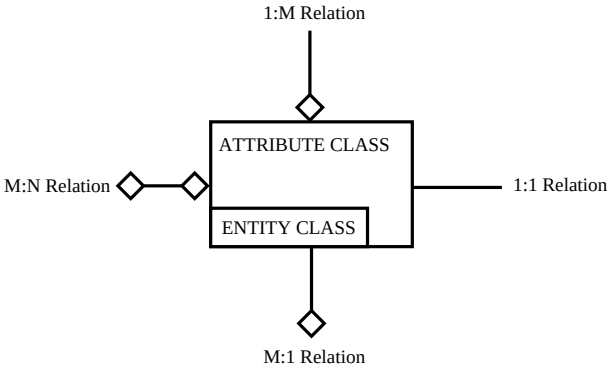
Relation Class Label – A phrase-like definition which reflects the meaning of the relationship expressed between two entities shown on the diagram on which the label appears.

Relation Ratio – The relation property which “loosely” defines how many relationships may exist for each member of an entity class.

Source(s) – One of the members of the functional organization whose responsibility it is to provide the elements of information (documents, forms, procedures, knowledge, etc.) on which the development of the model will commence and continue.

Validation – An effort which results in the informed consensus of the experts who are knowledgeable about the model; the model is considered “valid” if the majority of experts agree that it appropriately and completely represents the area of concern.

Section 9.0 IDEF1 Index of Terms



IDEF1 Index Of Terms

Appendix A

IDEF Family of Methods Overview and

Practical Guidelines for IDEF Use

Richard J. Mayer, Ph.D.
Knowledge Based Systems, Inc.
One KBSI Place
1408 University Drive East
College Station, TX 77840
(409) 260-5274

Capt. Michael K. Painter
Armstrong Laboratory/ HRGA
WPAFB, OH 45433
(513) 255-7775

Paula S. deWitte, Ph.D.
Knowledge Based Systems, Inc.
One KBSI Place
1408 University Drive East
College Station, TX 77840
(409) 260-5274

> "It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage than the creation of a new system. For the initiator has the enmity of all who would profit by the preservation of the old institution and merely lukewarm defenders in those who would gain by the new ones" ––
>
> Machiavelli, "The Prince," 1513 AD.

Introduction

In today's environment, system implementors in Corporate Information Management (CIM), Concurrent Engineering (CE), and Computer Integrated Manufacturing (CAM) face two overwhelming challenges. Besides their primary responsibility for introducing a new system into their engineering and manufacturing organizations, they also have an underlying need to introduce a new system of processes for developing these implementations. These new system development processes must employ an integrated *framework* of modeling methods. That is, the development process uses a structured collection of methods, rules, procedures,

and tools to support the development and evolution of systems. The framework guides the user in applying the appropriate method within the system development life-cycle. The goal of this paper is to provide some insight into the purpose of modeling, particularly from the perspective of a CIM or CE project manager/engineer who must select, use, and evaluate the results of modeling efforts in support of systems development.

For all the rapid advances in computer hardware and specific software technology (e.g., databases), the bane of large-scale information systems development continues to be the lack of effective, well and widely understood methods for engineering such systems. With additional requirements for the system to be *integrated* and *evolving*, the complexities become truly overwhelming. A solid base of methods support is essential to these types of system development efforts. Complex, large-scale, evolving, integrated systems require multiple, diverse methods, each for a specific purpose. Thus, there is clearly a critical need for the development of effective methods. As a result, many methods remain to be developed.

A two-prong approach to method development is necessary. The goal of the first approach is to identify the methods needed to support an evolving, integrated information system (EIIS) and develop these methods. In the context of this paper, emphasis is placed on analysis and design methods needed to support EIISs for CE and CIM. Part of this method development includes developing precise, mathematical-based formalizations of the individual methods, as well as capabilities for translating among the methods.

The second approach is to develop an *engineering discipline* for the appropriate selection, use, extension, and creation of methods to support the planning, analysis, and design of large-scale, EIISs. That is, methods that are developed, in turn, are used recursively to develop new system engineering methods with predictable effectiveness. Within this approach, techniques for the analysis and comparison of methods must also be developed.

Developing methods (and particularly developing a methods engineering discipline) requires an uncommon experience base in method work. A method development team must include members with extensive experience in the actual application of the technology, as well as members with experience in methodization, language syntax design (both graphical and lexical), and formal models of semantics.

From a scientific viewpoint, the two interesting observations of methods appear to contradict each other. The first salient observation is that methods, almost by definition, are not derived in any logical, traceable manner; the second observation is that in spite of this, *they*

work. If the history of a method's development could be captured, it would be obvious that the method was not developed arbitrarily. Yet, methods do tend to be developed piece-meal, over time, by many individuals and as a result of a discovery process of what works well in a domain. Indeed, the underlying theoretical reasons for why a method works may not be understood at all by the discoverer. Rather, a hunch, an intuition, or an accidental circumstance may lead to such discoveries. For the very reason that a theoretical basis for a method may be unknown or not well understood, methods also tend to be difficult to enforce. It is simply difficult to convince someone to use a method when given no logical reasons, as the following anecdote illustrates.

> A young welder new to a railroad construction yard was assigned the job of forming and welding the stays for the large wooden barrels used as containers on the railroad cars. Proud of his welding prowess, the newcomer chafed that the foreman insisted on giving him instructions at great length on exactly how to carry out the job–instructions which the young welder considered outdated. He argued with the foreman that he could do the job much faster his own way. The foreman, for his part, insisted adamantly that the job must be done just this certain way or it wouldn't be done right.
>
> Unconvinced, the welder decided to show the foreman. In two days, he completed a week's worth of work which he triumphantly showed the foreman. Wordlessly, the foreman took one of the stays in hand, climbed to the top of the water tower, and threw the stay to the ground. The welder watched as the stay bounced twice and shattered at the seam. Climbing down from the water tower, the foreman quietly handed the welder a stay built the foreman's way and pointed to the tower. When the welder repeated the experiment with the foreman's stay, the stay hit the ground bouncing repeatedly, but holding firm.
>
> While the young welder may never understand why the foreman's method works, there is no doubt that he will use it religiously in the future. Whatever the exact nature of the foreman's particular method, it was typical of methods because it represented "best practice" in the domain of welding stays, a best practice learned over time and through experience. In fact, a method may be abstractly described as an encapsulation of best practice in a domain of cognitive or physical activity.

Nature and Importance of Methods

The purpose of a method is realized through its use by a human mind. Just as shovels themselves do not dig holes, but provide leverage for a human to dig holes, methods provide leverage for the human mind to accomplish a job more effectively. The method may assist

and motivate the intellectual activities of the human, but it doesn't make the decisions, create the insights, or discover the problems.

Recognizing the nature of a method as an enabler, and not as a creator, should not diminish the recognition of the importance of methods. As the anecdote in the previous section illustrates, it may not be easy to pass down the knowledge of best practice from the expert to the novice. The basic importance of methods has long been recognized in the manufacturing industry. Methods are prominent in the "5 M's" of manufacturing–manpower, methods, materials, machines, and money. Materials, machines, and even money can be replaced, but manpower and methods that leverage the knowledge of the manpower are vital components of industry.

Components of a Method

Informally, a method is thought of as a procedure for doing something. That is, methods attempt to capture the "best practice" or experience. In addition, the method may have a representational notation to communicate this procedure more effectively. More formally, a method consists of three components as illustrated in Figure A-1. Each method has a definition, a discipline, and many uses. The definition contains the concepts, motivation, and the theory behind the method. The discipline includes the syntax of the method, a computer interpretable format (labelled ISyCL [Mayer 91g] Syntax in Figure A-1), and the procedure governing its use. Many methods have multiple syntaxes which have either evolved over time or are used for different aspects. Perhaps the most visible component of a method is the language associated with the discipline. Many system analysis and engineering methods use a graphical syntax to provide visualization of collected data in such a way that key information is unambiguously displayed. The use of a method may be by itself or within a suite of methods.

Types of Methods

The methods in the EIIS context are primarily methods that produce *models*, but some methods produce *descriptions*. Models and descriptions are similar in that they both consist of diagrams and texts. A model can be characterized as an idealized system of objects,

properties, and relations designed in certain relevant respects within a particular structure to imitate the character of a given real-world system. The power of a model comes from its ability to simplify the real-world system it represents, and to predict certain facts about that system with corresponding facts within the model. Thus, a model is a designed system in its own right, constrained to satisfy certain conditions by the abstract system of which it is an instance.

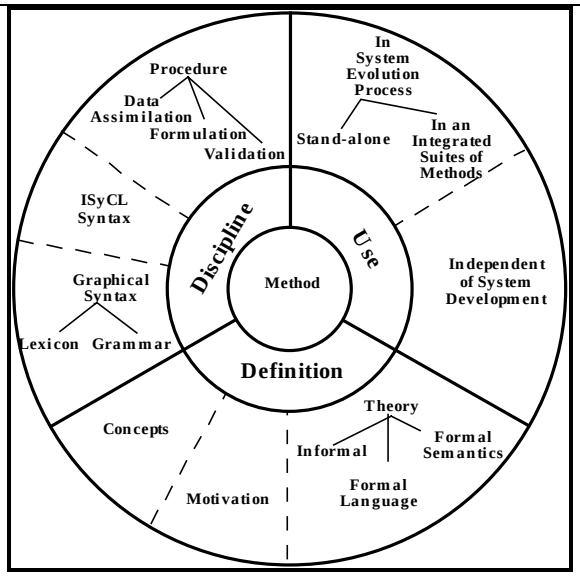


Figure A-1. Components of a Method

Models are known to be incorrect, but assumed to be "close enough" to provide reliable predictors of the real properties of the domain of interest. A description, on the other hand, is a recording of facts or beliefs about the world. As such, descriptions are, in general, partial. A person giving a description may omit facts that don't seem relevant, or that he has forgotten in the course of describing the system. Thus, while descriptions must be accurate, they are not constrained by abstract, testable conditions that must be satisfied.

It is important to distinguish conceptually between models and descriptions. Unfortunately, the term *model* may be used ambiguously in a general sense to mean both a description and a model. This occurs most frequently when discussing activities of modeling activities common to many methods and not a particular method. This paper discusses the family of IDEF methods including, descriptive methods, such as, IDEF3 and IDEF5 and modeling methods such as, IDEFØ and IDEF1. In this paper, the terms *model, modeler*, or *modeling* may refer either to models in the most general sense (referring to both models and descriptions) and models in the more restricted definition (i.e., versus descriptions). The context of the use of the term will clarify its intended meaning.

Methods in the System Development Process

All too often, modelers are caught between the system developer who claims that the effort cannot afford the lack of the modeling activity and the funding source who claims that the

modeling effort is prohibitively expensive. In fact, there are many reasons to justify the time, labor, and expense required by a modeling activity to build system models or descriptions. A model of the system development process can be used to illustrate where system modeling fits. Figure A-2 depicts the customer's perspective of the activities involved in developing a system and their relationships to one another. The nature of the system is unimportant and may be anything including night vision goggles, an information system, or even a house.

In the activity labeled *Establish Requirements*, the customer provides input in the form of an expression of needs. While the customer may recognize problems in the environment, he/she usually does not understand the exact cause and may, in fact, misdiagnosis a symptom. *Needs* are conditions that must be satisfied for the problem to be considered solved. The system developer must establish the limiting constraints on the conditions to be satisfied, (i.e., the *requirements*). For example, an architect tasked with designing a new home will take an expression of needs for more space and lower heating bills and will ask questions that determine requirements.

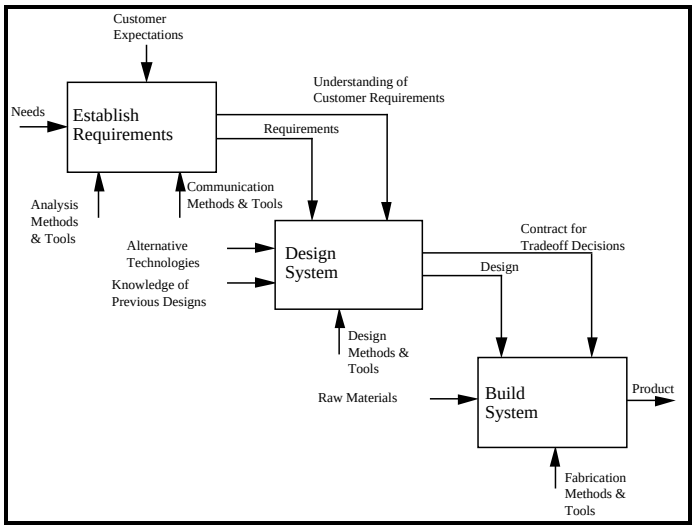


Figure A-2. Develop System (IDEFØ Model)

How big is your lot size?

What kind of additional space do you need? Storage space? Sleeping space?

What is an acceptable range for heating bills?

How much money is to be spent on this house?

This activity produces two results: 1) a clear set of requirements and 2) a perception of the customer that the environment is understood well enough by the developer that the customers' needs will be satisfied. This perception is a by-product of the developers' attempt to isolate the causes of the problems through careful study of the environment where the symptoms occur. This analysis will not only lead to the rediscovery of the problems known to the customer, but will also often result in identifying other existing or potential problems.

As far as the customer is concerned, the developer can use any analysis method and tool as long as it accomplishes the goal and promotes lower costs and faster turn-around time. In this context, tools are considered support (usually automated) for using the prescribed methods. In fact, much generated by the method may never be directly seen by the customer. However, periodically, it is necessary to communicate to the customer what the system developer is actually thinking. This may amount to nothing more than asking the customer "Is this what you mean?"

The second activity, *Design System*, will typically not proceed without both a clear set of requirements and the customer's feeling that his expectations will be satisfied. The system developer then uses whatever design methods and tools will best help him satisfy the requirements. There may also be cases where communication methods and tools are used by the system developer to communicate with the customer throughout the design activity. An architect, for example, may generate a set of blueprints from the requirements without needing to consult further with the customer. Cases where it is necessary to involve the customer further will most likely occur when trade-off decisions have to be made. The system developer then presents the effects of competing design decisions in terms of constraints (e.g., cost). The trade-off decisions made at this point can then be captured either implicitly by verbal agreement or explicitly by formal sign-off.

Once the design is accomplished and the trade-offs have been accepted by the customer, the building of the system can begin. The fabrication methods and tools to be used in this process can range in sophistication from strictly manual approaches to totally automatic system generation.

Models built as part of the system development process, as a minimum, should 1) instill in the customer a feeling of assurance that the system developer understands the customer's environment, the customer needs, and the conditions that must be satisfied to meet

expectations or the system requirements and 2) involve the customer in making trade-off decisions and document those decisions. One important purpose for modeling from the customer's perspective is to satisfy these needs. A corollary to this assertion is that if the developer is building models that do not satisfy these expectations, success in meeting the customer's needs is left largely to chance. This should in no way overshadow the importance of models to system development needs. Rather, these guidelines should be used to help guide what kind and how much modeling is actually needed.

With the purpose of modeling from the customer's perspective more fully understood, the discussion will center on experiences in the use of the IDEF (Integrated Computer-Aided Manufacturing (ICAM) DEFinition) methods to perform modeling activities in support of CIM and CE system development. Experience with three such methods will be described in detail, namely, IDEFØ Function Modeling, IDEF1 Information Modeling, and IDEF1X Data Modeling. Following this discussion, the emerging IDEF methods including IDEF3 Process Description Capture, IDEF4 Object-oriented Design, IDEF5 Ontology Description, and IDEF6 Design Rationale Capture will be introduced and their envisioned application potential for CIM implementations described.

Function Modeling Using IDEFØ

The IDEFØ Function Modeling method is designed to model the decisions, actions, and activities of an organization or system. IDEFØ was derived from a well-established graphical language known as the Structured Analysis and Design Technique (SADT) [Mayer 90]. The Air Force commissioned the developers of SADT to develop a function modeling method for analyzing and communicating the functional perspective of a system. Effective IDEFØ models assist in organizing system analysis and promoting effective communication between the analyst and the customer. In addition, the IDEFØ modeling method establishes the scope of analysis either for a particular functional analysis or for future analyses from another system perspective. As a communication tool, IDEFØ enhances domain expert involvement and consensus decision-making through simplified graphical devices. As an analysis tool, IDEFØ assists the modeler in identifying functions performed, what is needed to perform those functions, what the current system does correctly, and what the current system does incorrectly. Thus, IDEFØ models are often created as one of the first tasks of a system development effort.

Modeling Systems from an IDEFØ Perspective

IDEFØ includes both a process and a language for constructing a model of the decisions, actions, and activities in an organization. Applying the IDEFØ method results in an organized representation of the activities and the important relations between these activities in a nontemporal, non-departmentalized fashion. IDEFØ is designed to allow the user to "tell the story" of what an organization does. It does not support the specification of a recipe or process. Such detailed descriptions of the specific logic or timing associated with the activities requires the IDEF3 Process Description Capture Method. IDEFØ models isolate or separate *functions* from *organizations*, identifying common functional threads across organizational units, and facilitating organization-independent analysis.

IDEFØ has been successfully used as both an analysis tool and as a communication tool in a number of application areas. Referring back to Figure A-2, this characterization indicates that IDEFØ can be applied as a mechanism for performing the *Establish Requirements* activity. The communication facilitation capability of IDEFØ makes it an effective analysis tool for cooperative interdisciplinary team projects as those required by any CIM or CE initiative.

Organizational Structures and Strategies of IDEFØ

A number of organization strategies designed in the IDEFØ method lend tremendous expressive power and ease in communication. When improperly used or not understood, they yield models that are difficult to comprehend or may make absurd declarations which appear well-founded. Examples of IDEFØ organization strategies include 1) the purpose, viewpoint, and context statements, 2) the hierarchical or top-down analysis approach to model development, and 3) the levels of abstraction.

For the modeler, these organization strategies focus work on one piece of the model and establish clear boundary conditions within which to perform the analysis. For the customer, they allow rapid discovery and inspection of the pieces of the system with which the customer is most familiar. They also provide powerful browsing mechanisms for learning about the system as a whole and communicating the modeler's understanding of the system. The following will address the use of purpose, viewpoint, and context statements as organization strategies in the IDEFØ method. Following that discussion, the hierarchical or top-down analysis approach to model development and the notion of levels of abstraction will be discussed.

To begin an IDEFØ modeling activity, the modeler must first determine (and clearly describe) what the *purpose* of the model is, from what *viewpoint* the activity descriptions will be formulated, and within what *context*. The purpose is a statement of the goals of the modeling activities (e.g., what information needs to be assembled, what decisions this information is supposed to support, what consensus is to be achieved, etc.). For example, one purpose of an IDEFØ functional analysis could be to identify opportunities for consolidating existing functions under a new CIM strategy. An accepted purpose provides the modeling team with a completion criteria. That is, when the purpose is accomplished, the model is finished.

The viewpoint statement describes the perspective that should be taken when constructing, reviewing, or reading a model. This viewpoint establishes how the reader will interpret the model and how the modeler will constrain his idealization or abstraction of the activities that occur in the system under study. An accepted viewpoint statement provides the modeling team a mechanism for controlling the scope and level of detail in a model.

The context establishes the interpretation and scope of the model as part of a larger scope. This focus creates a boundary within the environment for the model.

Another strategy for organizing the development of IDEFØ models is the notion of *hierarchical decomposition* of activities. Although IDEFØ models are developed using a hierarchical or top-down approach, Doug Ross, the creator of SADT, is thought to have often struggled with these terms. In fact, Mr. Ross proposed that this process might more accurately be characterized as an Outside-in approach [Ross 85]. A *box* in an IDEFØ model, after all, represents the boundaries drawn around some activity. Looking inside that box leads one to discover the breakdown of that activity into smaller activities which together comprise the box at the higher level.

This hierarchical structure helps the analyst keep the scope of the model within the boundaries represented by the activity's decomposition. The customer also finds this organization strategy useful for hiding unnecessary complexity from view until a more in-depth understanding is required by looking inside the box at its decomposition (See Figure A-3).

This complexity hiding may also be characterized as part of the abstraction mechanism used in IDEFØ. One common misconception, however, is that levels of abstraction are only evidenced in the activities themselves as one moves between levels of the model. The arrows

also exhibit different levels of abstraction between levels of the model. In fact, achieving the correct balance between the level of abstraction associated with a box and the level of abstraction associated with the arrows attached to the box is not always trivial. For example, suppose the four mechanism arrows inside the inner box in Figure A-3 represent different types of tools used to accomplish their respective activities. These four arrows could be bundled together into a more abstract perspective as a single arrow labeled "tools." Thus, the outer box in Figure A-3 would have only one mechanism arrow for its level of abstraction. A far less elegant depiction would have all four arrows appear at both the more abstract and the more detailed levels of the model. It can be seen that one easy way to tell whether or not the modeler has effectively used the information hiding constructs available in IDEFØ is to count the number of arrows attached to the boxes at any given level. If the model seems cluttered with arrows, it is very likely that the level of abstraction used in bundling the arrows is not the same as the level of abstraction as the activity.

Perhaps the least understood and most frequently misapplied IDEFØ constructs for screening unnecessary detail at a given level of abstraction is the notion of bundling and unbundling of arrows. It would be logically inconsistent to unbundle all but two of the four mechanism arrows in Figure A-3 when each occurs at the same level of abstraction, namely, as component tools. Likewise, it would make little sense to unbundle and rebundle an arrow at the same level of abstraction. Unfortunately, the IDEFØ literature does not adequately cover how to appropriately avoid logical inconsistencies that can be introduced through incorrect use of information hiding constructs through arrow bundling. The best approach is to build models using an automated support tool that enforces good practice. Otherwise, an IDEFØ modeler can take years to learn how to recognize and avoid bundling problems.

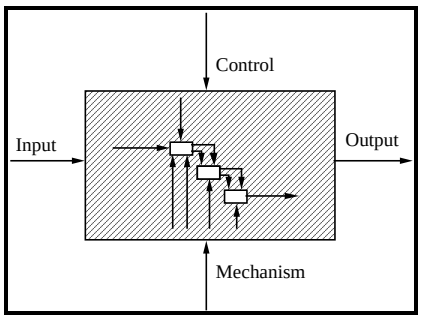


Figure A-3. Looking Outside-In

IDEFØ Guidelines

There is a considerable knowledge base of heuristics available for aspiring IDEFØ modelers [Cullinane et. al. 90, KBSI 90a, Mayer 89b, Softech 81a]. In the following paragraphs, a few of the more common hints for obtaining the most from an IDEFØ modeling exercise are discussed. Specifically, these guidelines will help determine how well a given model satisfies the purposes of modeling outlined above.

Without question, the most difficult aspect to master in IDEFØ modeling is maintaining the same consistent purpose and viewpoint between levels of the model. What makes this task so difficult is the difficulty of recognizing shifting viewpoints. A good heuristic is to look at the boundaries within which the modeler develops a decomposition and formulate questions including the following: "Does this activity fall within the scope of the higher-level activity?" and "Does this activity conform to the established viewpoint and purpose of the model?"

Secondly, look for models that push the numerical constraints of the method. For example, the discipline component of the IDEFØ method establishes a rule that there should never be less than three nor more than six activities to a decomposition. Likewise, there should never be more than six arrows on one side of an activity box. A common modeling mistake is to decide that a seventh activity in the decomposition becomes indispensable. In fact, one tendency is first to draw six boxes for the decomposition and then attempt to come up with names for all six activities. Another mistake occurs when activity boxes and their associated arrows begin to look like wiring diagrams for electronic components. This is largely due to a failure to logically organize the arrows into bundles at different levels of abstraction consistent with the same level of abstraction as their associated activity boxes. Inappropriate bundling, such as that done simply to abide by the established rules, may also occur. These kind of errors become obvious when arrows seem arbitrarily grouped together.

Another common problem emerges when new conventions are introduced into the method. For example, some modelers will choose to establish a convention that inputs and outputs can only be data elements. In this way, they hope to ensure that inputs and outputs translate directly into information model elements. Intuitively, this approach would then provide clear and unambiguous tracking of data needs across activities as well as clearly delineating the scope of their information models. However, with this approach, the modeler is forced to further change conventions by making mechanisms into resources assigned to an activity. For example, consider an activity *Fix Broken Airplane*. With these conventions,

there is only one place for the broken airplane, as an input to the activity. But with this approach, there is no place to show that what emerges is a fixed airplane, since outputs can only be data elements. It is far more useful to use the existing conventions and then use inputs and outputs as candidates for what may be data elements to be examined and discriminated later.

A frequently misunderstood convention of IDEFØ is the built-in notion of nontemporality implicit in the IDEFØ method. As discussed previously, IDEFØ does not explicitly capture time-ordered constraints between activities. In fact, temporal logic is purposely not included in IDEFØ to lend more expressive power and generality. While IDEFØ could provide a description of a specific set of activities operating within the bounds of a specific time-ordered process, it is far more useful for analysis purposes to provide a generalized model which accounts for all, or at least all relevant, paths that could be taken through a set of activities for any number of time-ordered processes. Models should therefore be judged, in part, by the degree to which they accommodate likely or possible sequences of activities and should not be built with the intention of implying a specific process.

Perhaps the most useful exercise used for assessing the quality of an IDEFØ model is to sit down, read the model, and determine if it makes sense with a constraint of no more than two minutes per page. For a large model, this should require no more than two hours. If a reader can understand the environment modeled by the IDEFØ representation within that time and feel capable of explaining what occurs in that environment, it is likely that the model is of significant value. Models that require two full days of careful study to fully comprehend are not good IDEFØ models.

Information Modeling Using IDEF1

Referring to Figure A-2, IDEF1 is viewed as a method for both analysis and communication in establishing requirements. In this case, however, IDEF1 establishes the requirements for what information is or should be managed by enterprise. In CIM applications, IDEF1 is generally used to 1) identify what information is currently managed in the organization, 2) identify which of the problems identified during the needs analysis are caused by lack of managing appropriate information, and 3) specify what information will be managed in the "TO-BE" CIM implementation.

The IDEF1 information modeling method derives its foundations from three primary sources. The Entity-Link-Key-Attribute (ELKA) method developed by Hughes Aircraft, the Entity-

Relationship (ER) method proposed by Peter Chen, and Codd's Relational Model (see Figure A-4). The original intent of IDEF1 was to capture what information exists or should be managed about objects within the scope of an enterprise. The IDEF1 perspective of an information system includes not only the automated system components, but also non-automated objects such as people, filing cabinets, telephones, etc. IDEF1 was specifically designed to not be a database design method. At the time of IDEF1 development, the database community believed that a method for analyzing and stating information resource management needs and requirements was needed. This was the intent of IDEF1. Rather than a design method, IDEF1 is an analysis method used to identify the following.

1. The information collected, stored, and managed by the enterprise.
2. The rules governing the management of information.
3. Logical relationships within the enterprise reflected in the information.
4. Problems resulting from the lack of good information management.

The results of information analysis can be used by strategic and tactical planners within the enterprise to leverage the information assets to achieve competitive advantage. Part of these plans may include the design and implementation of automated systems which can more effectively take advantage of the information available to the enterprise. IDEF1 models provide the basis for those design decisions. IDEF1, then, is not used to design a database; rather, it is used to provide managers with the insight and knowledge required to establish good information management policy. The next section will provide an overview of some of the basic concepts and rules of IDEF1. The interested reader is referred to [Softech 81b, KBSI 90b, Mayer 89b, Menzel 89] for additional details regarding this method.

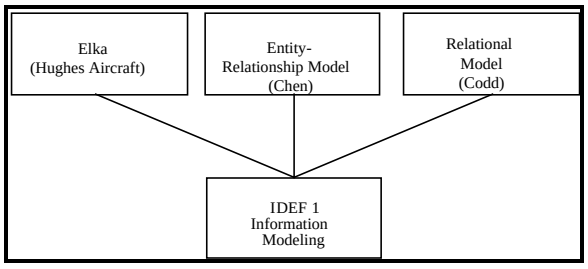


Figure A-4. IDEF1 Origins

Modeling Systems From The IDEF1 Perspective

IDEF1 uses simple graphical conventions to embody a powerful set of rules which help the modeler distinguish between the following.

1. Real-world objects.
2. Physical or abstract associations maintained between real-world objects.
3. The information managed about the real-world object.
4. The data structure used to represent that information for acquiring, applying, and managing that information.

Simply stated, IDEF1 was designed to represent information that is, or should be, collected, managed, controlled, and ultimately paid for by the enterprise (item 3). The rules of the method help prevent modeling items 1 and 2 (normally considered the province of knowledge engineers). They also divert the attention of the modeler away from database design (item 4, normally considered the province of software engineers).

There are two important realms to modelers in determining information requirements. The first realm is the real-world as perceived by the people in an organization. This realm includes the physical and conceptual objects (e.g., people, places, things, ideas, etc.), the properties of those objects, and the relations associated with those objects. The second realm is the information realm. This realm includes information images of those objects found in the real-world. An information image is not the real-world object, but only the information collected, stored, and managed about real-world objects. IDEF1 is designed to assist in discovering, organizing, and documenting this information image. These tasks are essential to any CIM implementation. This does not mean that the task of structuring the organization's knowledge of the first realm is not important. Looking towards intelligent CIM systems and/or CIM implementations that embody large numbers of knowledge based systems, this task (often referred to as *ontology* definition) becomes more important. However, a specialized method, IDEF5, is designed to assist in the structuring of this knowledge base. IDEF5 and its application are discussed in a later section. The following section concentrates on the role of IDEF1 which attempts to capture an organization's information management requirements.

IDEF1 Basic Concepts

Focus on the real-world realm for a moment (see Figure A-5). The term *real-world object* is used to describe real-world people, places, things, or ideas. In this sense, the sales

department in a company may be a real-world object, as is an employee working in that department. These objects have characteristic properties associated with them, such as a name, age, gender, etc. Further, one real-world object may be involved in some kind of association with other real-world objects. For example, an employee may *work for* a department.

Now, focus on the information realm. An IDEF1 *entity* represents the information maintained in a specific organization about physical or conceptual objects (e.g., people, places, things, or ideas). For example, an IDEF1 entity exists when an organization maintains information about the sales department resulting in the existence of an *information image* of that object in the organization's information system. In IDEF1, the term *entity class* refers to a collection of entities or the class of information kept about objects in the real-world. An *entity class* can be thought of as an empty box for holding 3" x 5" cards with each card an actual entity. The box is labeled on the outside with 1) an entity class name describing what type of cards go in the box, and 2) a template for the individual cards that will eventually go inside.

Entities have characteristic *attributes* associated with them that record values of properties of the real-world objects. Using the card file model, an *attribute class* is the template for the attribute-value pairs found on the individual file cards. The term *attribute class* refers to the set of attribute-value pairs formed by grouping the name of the attribute found on the outside of the file box, and the values of that attribute for individual entity class members (entities), listed on the individual cards themselves. A collection of one (or more) attribute classes which allow us to distinguish one card from another, or one member of an entity class from another, is called a *Key Class*. A Key Class is indicated by placing it in the top left corner of the template and underlining it.

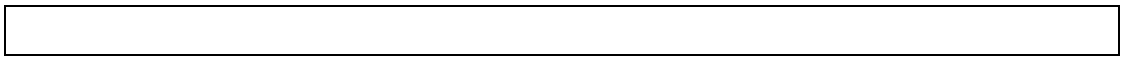
A *relation* in IDEF1 is an association between two individual information images. The existence of such a reference is discovered (or verified) by noting that the attribute classes of the one entity class contain the attribute classes of the Key Class of the referenced entity class member. For example, the information managed about an employee may contain a department number (an attribute class that belongs to the collection of information kept about a department). A *relation class* can be thought of as the template for associations that exist between entity classes. An example of a relation class in IDEF1 would be the label *works for* on the link between the information entity class *employee* and the information entity class *department*. It is important to note that if no information is kept about an

association between two or more objects in the real-world, then from an IDEF1 point of view, no relation exists. Relation classes are represented by links between the entity class boxes on an IDEF1 diagram. The diamonds on the end of the links and the half diamonds in the middle of the links encode additional information about the relation class (i.e., cardinality and dependency). These links often indicate the existence of a business rule of an organization. If there are inconsistencies during the analysis of these links, the analyst very often discovers inconsistencies in business rules.

The procedure portion of the IDEF1 method was designed to be scalable from small department level analyses to large enterprise-wide projects. It has been demonstrated as an effective problem solving method where the cause of a particular problem has to do with the lack of management (or mismanagement) of information. However, it is important to have the analysis done correctly. The next section describes some indicators of quality in an information analysis performed using IDEF1.

IDEF1 Modeling Guidelines

There are some very simple ways to quickly inspect an IDEF1 model and determine whether or not the modeler has been able to take the expression of need, together with an understanding of the environment, and successfully identify existing and future information management requirements. These inspection techniques will help the customer identify, almost immediately, whether or not the IDEF1 modeler is modeling from the real-world realm (generally considered incorrect) or from the information realm (generally considered correct).



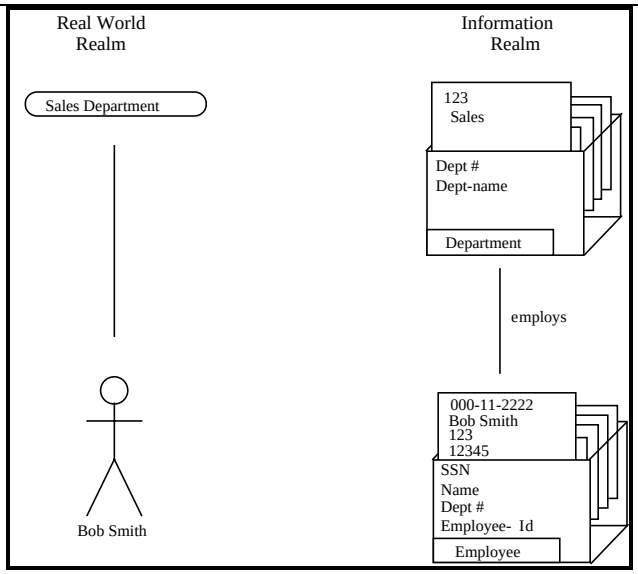


Figure A-5. IDEF1 Basic Concepts

First, check the labels used to name the entity classes. If the labels are plural rather than singular names, it is unlikely that the model was created with the information realm in mind. For example, a box labeled *Employee* conforms better to the convention that the entity class represents the set of information important to manage about a real-world person than a box labeled *Employees*. With the *Employees* label, the modeler may mean a group of people with some of the same properties (i.e., individuals who all have a job at the company). Remember, an IDEF1 modeler is not concerned about the real-world objects directly, only the information about them that is actually managed by the organization.

A second indication of models which represent real-world objects and not information images are models which have some boxes without non-key attribute classes and others with whole laundry lists of them. Those models without non-key attribute classes typically are objects like organizational units. Close inspection of boxes with long lists of attributes will often reveal that the box represents a form, like a *Purchase Order* form, with the attributes of the actual fields on the form.

The third check is to look for models that have most of their entity boxes with only one or two relations whereas a few entity boxes are wired with many relations. In this case, it is highly probable that the box with all the relations will be the entity box with a label such as *Person*. In this example, many of the relations attached to the enetity box indicate the roles that a

person can assume. For example, the model might read: Person inspects Parts; Person certifies Inspection; or Person is married to Person.

Although it is a natural tendency for first-time IDEF1 users to model what they know about the real-world in terms of what can be easily seen, misapplication of IDEF1 may lead to huge models with little or no benefit. With little direction, the novice modeler attempts to model everything that is observed in the real-world. These models become more and more difficult to manage as they continue to grow. Soon, it becomes easy to lose sight of the purpose of the modeling exercise. Information models that model the real-world realm rather than the information realm provide virtually no insight into what the information requirements are or where more effective information management policies can improve competitive posture.

Guidelines When Using IDEF1

A potential trap for information system developers is to assume that the ability to uniquely identify one information entity from another necessarily implies that one real-world object can be uniquely identifiable from another. Remember, an information model represents information that is actually managed about real-world objects. Any other knowledge is unavailable to the information system. When information is kept about individual real-world objects, the information model can be used as a means for identifying specific real-world objects. For example, the serial number used to distinguish one engine from another may also serve to distinguish one entity called *Engine* from another. However, when information is kept only about types of real-world objects, the same situation does not hold. Consider, for example, a standard bolt. A unique part number is not usually assigned to each bolt. The information model, however, may use part number to uniquely identify the information kept about specific families of bolts.

To avoid confusion , the IDEF1 modeler should be careful naming the relation classes and the links that connect entity class boxes. As seen in Figure A-5, it seems very intuitive to read the model as, “A Department employs one or more Employees.” Is this the same as, “The information kept about Departments employs the information kept about Employees?” Obviously, the first reading may confuse one into thinking that the file box is actually the real-world object and not the information kept about an object. Remember, the link between the boxes represents a reference from one file card to another, or information relationships, not real-world relationships.

A sentence such as “A Department employs one or more Employees” is a natural language fact or a business rule. Since business rules are important to know and manage, perhaps they should be tracked and managed in a similar fashion to show how source data items in IDEF1 are tracked and managed. The only exception being that they are managed separately as a source facts list. In the meantime, the links between entity classes can be labeled L1, L2, etc. This makes it very clear that a relation class represents a function which when applied to a member of one entity class will return those entity class members involved in the information relationship. Information relationships and real-world associations cannot be indiscriminately mixed without leading to confusion, paradox, and error [ISO 87]. For example, suppose an entity class *System* is created and attached to a relation originating from and ending at the *System* entity class with a *is comprised of* relation class label. How is such a model interpreted?

Intuitively, it seems reasonable to interpret the entity box as representing a system/subsystem hierarchy. The highest level system may be something like an automobile that is comprised of subsystems and further subsystems to the individual component level. With this approach, it appears that the same system/subsystem hierarchy can be ascribed to the information kept about a vehicle to enable more intuitive thinking. This approach implies that the information kept about the highest level system is comprised of the information kept about subsystems below it in the system hierarchy. However, this doesn't work because the information kept about an automobile includes information such as who owns it, what state it is registered in, what year it was manufactured, etc. None of that information has anything to do with the information kept about subsystems such as the braking system, which might include the type of front and rear brakes, the moisture content in the brake fluid, or the thickness of the remaining brake pad. Obviously, information relationships will not behave in the same manner that real-world relationships behave. The only correct interpretation from an IDEF1 viewpoint is that one member of the class of information kept about an object can reference information about other objects through the referenced member's Key Class.

The rules and procedures of IDEF1 assist the construction of accurate models of information currently managed in an organization targeted for CIM implementation. They also serve as mechanisms for definition of the information requirements for the TO-BE system. However, IDEF1 was designed to be technology independent. Therefore, when the design for the CIM implementation is being started, two other IDEF methods are used: IDEF1X for relational

data base implementations and IDEF4 for object-oriented implementations. These methods are described in the following sections.

Data Modeling Using IDEF1X

In Figure A-2, IDEF1X is intended as a method for accomplishing the *Design System* activity. Because it is a design method, IDEF1X is not particularly suited to serve as an "AS-IS" analysis method, although it is often suggested as an alternative to IDEF1. IDEF1X is most useful for logical data base design after 1) the information requirements are known and 2) the decision to implement using a relational database has been made. Hence, the IDEF1X perspective of the information system is focused on the actual data elements in a relational database. If the target system is not a relational system, e.g., an object-oriented system, IDEF1X is not the best method. The interested reader is referred to [GE 85, KBSI 90c, Mayer 89b] for additional details regarding this method.

The development of IDEF1X was influenced by Chen's Entity Relationship (ER) model, Codd's Relational model, and Smith's Aggregation/Generalization model. These origins led to the development of the Logical Database Design Technique (LDDT) by the Database Design Group, Inc. LDDT later became a commercial product of Dan Appleton Company (DACOM) as the Data Modeling Technique (DMT). A coalition of companies lead by General Electric including SDRC, CDC, and DACOM used this method and their experience with IDEF1, to create the IDEF Data Modeling Method, or IDEF1X (See Figure A-6).

Modeling Systems From The IDEF1X Perspective

Once there is a thorough understanding of the information requirements, decisions about how to manage that information more effectively can be made. One possible decision is to implement an automated system, requiring the selection of an appropriate design method. Good design methods should result in a robust, high quality, cost effective implementation with affordable life-cycle costs. Therefore, a good design method must embody the best application experience associated with a particular technology. This usually means that a design method will generally work "well" only for that technology whose experience base it encapsulates. The IDEF family of methods recognizes this fact and provides specific methods for design with particular implementation technologies in mind. If the technology of choice is the relational database technology, IDEF1X is appropriate for logical database design. If the technology of choice is an object-oriented database paradigm, IDEF4 (discussed in a later section) is more appropriate.

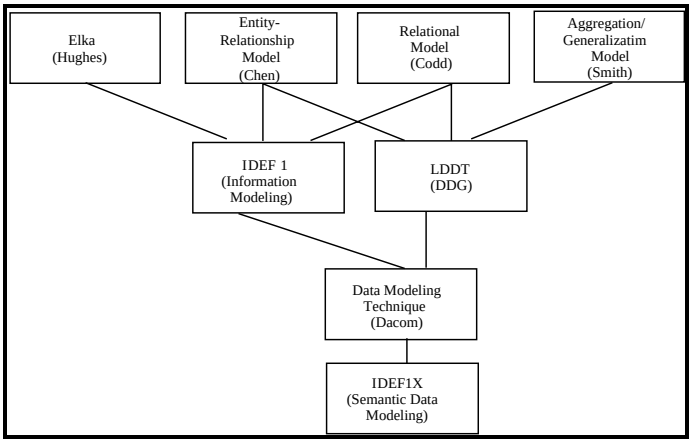


Figure A-6. IDEF1X Origins

There are several reasons why IDEF1X is not well suited for non-relational system implementations. IDEF1X requires that the modeler designate a Key Class to distinguish one entity from another, whereas object-oriented systems do not require keys to individuate one object from another. In situations where more than one attribute or set of attributes will serve equally well for individuating IDEF1X entities, the modeler must designate one as the primary key and all others as alternate keys. The explicit labeling of a foreign key, which is an attribute owned by one entity, but which serves as the key attribute in another entity, is also required. Modeling constructs like these clearly indicate specific intent to incorporate the best practice in logical design for relational systems implementations. The results (logical design models) of an IDEF1X activity are intended to be used by the programmers involved in taking the "blueprint" for an information system, or the logical database design, and implement that design in a relational database. However, the modeling language of IDEF1X is sufficiently similar to that of IDEF1 that the designs generated from the information requirements specification can be easily reviewed and understood by the ultimate users of the proposed system.

IDEF1X Basic Concepts

Since the terminology used in the IDEF1X method is very similar to that used by the IDEF1 method, further definition of terms is necessary to avoid confusion. There are fundamental differences in the theoretical foundations and concepts employed by the two methods. An *entity* in IDEF1X refers to a collection, or set, of similar data instances (data records about

persons, places, things, or events) that can be individually distinguished from one another. Thus, an entity box in IDEF1X represents a set of data items in the real-world realm. An *attribute* is a slot value associated with each individual member of the set, (these individual members are called *entity instances*). In Figure A-7, the record labeled *Bob Smith* and *Sales Department* are both entity instances. *Department* is the collection of specific records in a relational table representing departments; *Employee* is the collection of records about people employed by individual departments. *Department-Name*, *Department Number*, and *Department-Location* might be attributes of the *Department-Entity*. The *relationship* that exists between individual members of these sets is given a name. In this case, this relation is interpreted as establishing a *referential integrity constraint*.

A powerful feature of the IDEF1X method is the support for modeling logical data types through the use of classification structures. This classification structure is the *generalization/specialization construct*. This construct is an attempt to overlay models of the natural *kinds* of things that the data represents whereas the boxes, or entities, attempt to model *types* of data things. These *categorization relationships* represent mutually exclusive subsets of a generic entity or set. In other words, subsets that emerge from the superset cannot have common instances. One example of how this is used is to state that given a generic entity *Person*, two subsets can be created which represent a complete set of categories of people, namely, *Male* and *Female*. No instance of the *Male* set can be a instance of the *Female* set, and vice versa. The unique identifier attribute for an instance of the male set is, by definition, the same type of data as that for an instance of the generic entity. The same holds true for the female category entity. The general attributes that apply to all members of the entity *Person* are listed in the generic entity. The specialization attribute, gender in this case, is listed in the category entity.

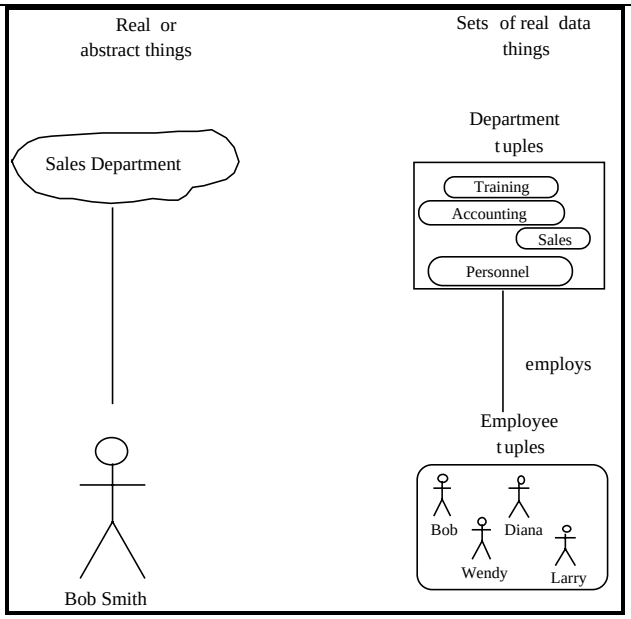


Figure A-7. IDEF1X Concepts

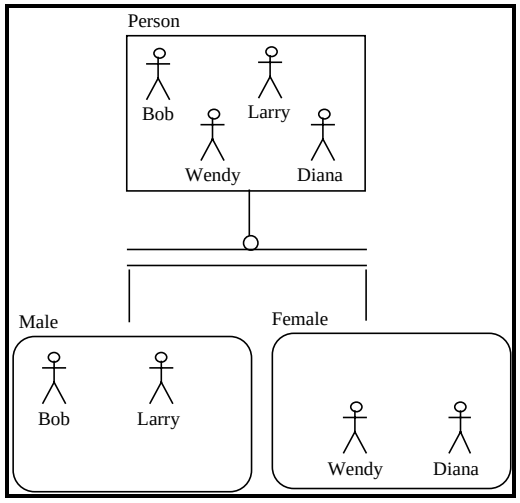


Figure A-8. IDEF1X Generalization/Specialization Construct

Guidelines When Using IDEF1X

The underlying concepts associated with the IDEF1X method are intended to bridge the modeling of natural language facts about real-world things, people, places, events, etc., with the modeling of logical data structure. This is very different than the IDEF1's goal which

focuses strictly on the information image of the real-world things (not the things themselves, nor the data structures that represent the information about the things). IDEF1X is often extended to go a step further by also attempting to design data specifications associated with these sets of data items. This is accomplished through the glossary associated with the attributes and relationships.

Considering the examples discussed in the section on IDEF1, the intent of IDEF1X is to model how the database represents facts such as "A department employs one or more employees" or "A system is comprised of one or more subsystems." Since real-world relationships between objects do not behave the same way that information relationships do, it is questionable if this approach is useful for designing logical data structures. The reader may want to address this question by reconsidering the example of the model representing hierarchies of systems. From considering this example, one can easily recognize ample possibility for misuse of the IDEF1X capabilities exists. This misuse results in models of the real-world things, and not sets of relational database entities.

Another example of this problem is illustrated in an entity called "Technical Order Improvement" that appeared in a model delivered to the Air Force intended to establish the specification for a logical database design. This entity name would have been more accurate with the word *Form* appended to the end of the entity name. This is because the attributes listed in the entity actually correspond to individual fields on an Air Force form called the AFTO 22. The entity here is used to model an artifact (namely the form itself) which is a list of attributes.

The first question to ask is, "Is this a reasonable model of the set of forms called AFTO 22s?" This entity certainly presents an accurate model of the fields on the existing form and may therefore be fully justified. The next question to ask is, "Has any design activity been facilitated through modeling the form in this manner?" In other words, "Does this kind of modeling assist in moving from a set of information requirements to the logical design of relational tables and from which trade-off decisions can be made?" Does it make sense to create a table, or relation, in a relational database that corresponds directly with what one would find on existing forms? This approach may have little or no impact for small implementations where only a limited set of artifacts have to be modeled. However, for large-scale implementations, the plethora of different forms would easily employ an army of modelers maintaining the inventory of the forms and their associated fields. More importantly, if such a model is handed to an implementor as the design specification, the

implementation will be a simple computerization of the current paper system. Such automation of paper systems are infamous in the history of CIM failures.

Based on these examples, it is obvious that misapplication of a design method can lead to confusion and/or poor design. One possible solution is to establish the convention that an entity must represent information kept about real-world objects rather than the objects themselves. But, this would result in little more than a syntactic variant of IDEF1, with one major exception, namely, the generalization/specialization construct. This too will lead to confusion if used without exercising great caution. For example, is it true that the set of information kept about people is divided into two types, namely, male information and female information? Probably not, and if so, it would probably be illegal. More importantly reducing IDEF1X to IDEF1 loses a good design method.

Perhaps, IDEF1X is best used as a method for modeling just the objects themselves and not the structure of the data associated with those objects. Models of this type are often called *concept models*. Again, the difficulty occurs with the generalization/specialization construct since in the real-world, kinds are usually overlapping, and not mutually disjoint. Since IDEF1X does not allow for categorization entities with mutual members, IDEF1X could not serve well as a concept modeling language. For example, a model of the kinds of employees found in a company might include managers, engineers, designers, and secretaries, among others. A generalization entity in this case would be *Employee*, and the specialization entities would be those previously listed. The interpretation of this structure implies that engineers cannot be designers, nor can they be managers.

Methods are needed for three different aspects. First, methods are needed that can effectively capture what is known about the real-world and the relationships that exist between people, places, events, etc. Second, there is a need for methods that can capture existing and anticipated information management requirements. Third, good methods for supporting the design of systems that apply specific technology to meet the information management requirements are needed. The IDEF3 Process Description Capture and IDEF5 Ontology Description methods discussed in the following sections are specifically designed to address the first of these needs. The IDEFØ Function Modeling and IDEF1 Information Modeling were specifically targeted at the second of these needs. The IDEF1X Data Modeling, IDEF2 System Dynamics Modeling, and IDEF4 Object-oriented Design were developed to satisfy the third need. Unfortunately, there are a number of commonly used

modeling languages which fail to maintain an unambiguous distinction between these three realms.

System Dynamics Modeling With IDEF2

The potential for benefits from the use of simulation modeling for manufacturing and information system analysis, planning, and design have been well established over the past 20 years. As a result, more and more decision makers turn to simulation to study the complex interactions and the dynamic behaviors of integrated manufacturing systems. Simulation modeling is a powerful decision support tool that aids in solving complex problems in a variety of application domains. Simulation is useful when:

1. The cause of a problem is the result of a complex time dependent interaction among the components of the system.
2. The effect of a change to an existing system needs to be analyzed.
3. A proposed system does not exist.
4. Options to improve or measure system performance need to be quantified.
5. Other quantitative analytic methods are computationally intractable.

Simulation allows one to ask "what if" questions and to derive new information from existing knowledge. The simulation activity, coupled with the evaluation of alternate designs and courses of action, can lead to a better understanding of system operations and management policies.

The widespread use of simulation as an effective manufacturing or system development decision aid has been thwarted by the requirement for extensive training and skill in the design and use of the simulation modeling technique. The frustration of simulation has been that domain experts who know how their systems operate, and who can describe in detail the system operation, have been unable to take advantage of simulation modeling. These experts have relied on experienced simulation analysts to design and develop the simulation model. This dependence on experienced analysts by the domain experts has made effective communication between these two parties imperative. The success of simulation activities depends on 1) how well the expert can describe the system to a simulation expert, 2) how well the simulation analyst can understand that system, and 3) the effectiveness of accurately translating systems descriptions and goals to a simulation model. IDEF2 is focused on

improving the productivity of the simulation modeler by improving the ability of the simulation modeler to communicate model assumptions and designs to the domain expert.

Thus IDEF2 is a simulation model design tool that provides a language for representing models of the time varying behavior of resources in a manufacturing system, providing a framework for specification of math model based simulations. IDEF2 was designed to improve the process of design of representative simulation models that can be executed to predict what a system will do under specific conditions. The interested reader is referred to [Softech 81c] for additional details regarding this method.

Simulation Model Design from the IDEF2 Perspective

The IDEF2 method development was based on an extensive experience base with continuous, discrete, and network simulation languages design and the application of these languages to industrial problems. The primary designers of IDEF2 were A. Alan B. Pritsker, Robin J. Miner, and John F. Ippolito of Pritsker & Associates, one of the leading industrial simulation companies in the United States. IDEF2 decomposes the design of a simulation model into the following four submodels.

1. Facility Submodel (used to specify the model of the agents of the system and environment).

2. Entity Flow Submodel (used to specify the model of the transformations that an entity undergoes).

3. Resource Disposition Submodel (specifies the logic of agent assignment to entity transform demands).

4. System Control Submodel (specifies the effect of transformation independent or system external events on the modeled system).

One of the goals of this submodel decomposition was the need to allow teams of analysts to easily partition the tasks associated with the construction of large models. It was also the intent that the submodel specifications could be reused as actual system specifications, e.g., use of the facilities submodel as a basis for other quantitative plant layout analysis, or the use of the resource disposition submodel and the system control submodel as the control logic specification for the shop floor control specifications. Finally, through the use of decomposition of the specification by behavioral partitioning, it was hoped that reuse of model specifications could be achieved.

IDEF2 Basic Concepts

IDEF2 is a graphical specification language with a computable interpretation. This means that simulation model designs are specified with a graphical syntax. However, they are complete enough to allow direct execution of the simulation model that they specify. This paper will not attempt to describe each element of the language in this brief summary, but rather illustrate each of the described submodels in the following figures.

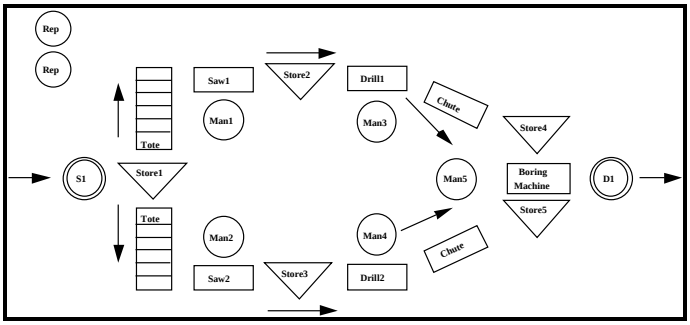


Figure A-9. Facility Submodel used for Plant Layout

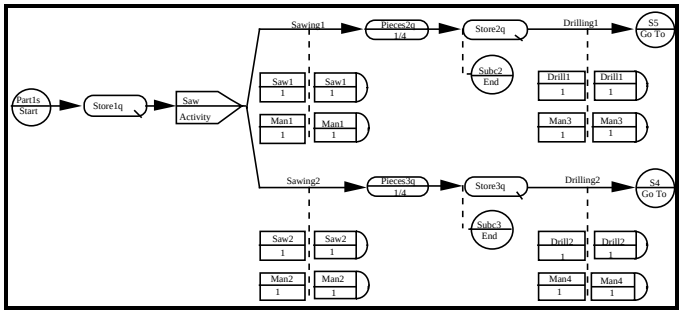


Figure A-10. Example Entity Flow Network

Issues In IDEF2 Modeling

IDEF2 was widely used in CIM implementation initiatives and factory modernization efforts. A VAX-based simulation and decision support system developed by Pritsker & Associates was used to analyze these models. At this time, the IDEF2 method is relatively dormant. Many of the specification capabilities, and graphical innovations have been incorporated into commercially available simulation languages (e.g., MAP, SLAM, and SIMAN). IDEF2 demonstrates the ability to reduce the semantic gap between simulation model design and an executable simulation program. It represents an important advance for improving the productivity of simulation modelers, but does little to aid the non-simulation trained decision maker. This is similiar to a traditional CAD system that aids a product designer in quickly producing specifications for a mechanical part, but provides no support for the actual design decisions behind those specifications.

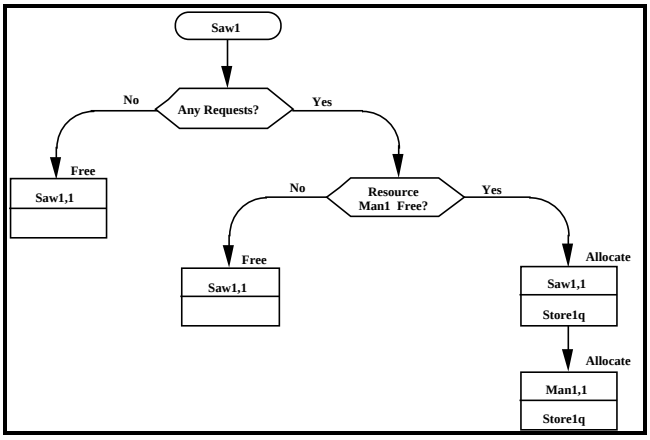


Figure A-11. Example Resource Disposition Submodel

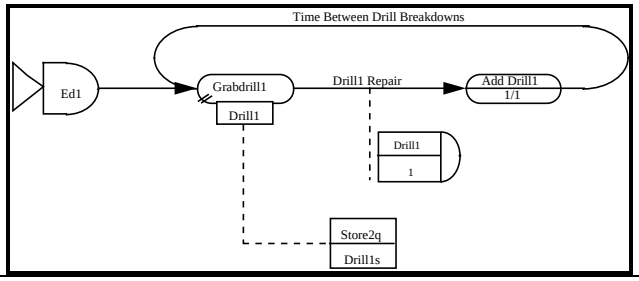


Figure A-12. Example System Control Submodel

The IDEF3 method (described in the next section) is targeted at assisting the domain experts in recording their knowledge about process flow and object state transitions in their environment. Continuing research is being directed at the construction of knowledge based systems to automate the design of simulation models from these IDEF3 process descriptions and the analysis question to be answered. Such a system would provide support for the decision maker to input what is known about the system (a description of how his system works, and a question about his system that he wants answered). In these environments, IDEF2 can provide an intermediate representation of the generated model design for review by the simulation analyst. It is expected that IDEF2 will be updated to incorporate object-based (if not object-oriented) specification constructs that have recently emerged.

IDEF: The Next Generation

A method is a designed system. Unlike other CIM technology, a method is designed to execute on the human mind (more often on a multidisciplinary collection of minds). Like any other system, a method extended beyond its designed limits will fail. The goal of the IDEF developments has been to provide an interlocking framework of methods for the definition (or reverse engineering), design, development and maintenance of information integrated systems. The term *framework* [Zachman 87, IUG 90, Mayer 90] means a structured collection of methods, rules, procedures, and tools to support the development and evolution of systems. Experience during the first seven years of applying the IDEF methods resulted in the identification of a number of additional method needs [Mayer 87] including the following.

1. Need to capture scenarios of logical or temporal sequences of events.
2. Need to design effective object-oriented applications and databases.
3. Need to capture reference descriptions of the objects described in the real-world realm.
4. Need to record CIM design decision rationale in order to achieve TQM on the CIM system itself.

These needs are being addressed by the next generation of IDEF methods, described in the following sections.

Process Flow & Object State Descriptions With IDEF3

One of the most common communication mechanisms to describe a situation or process is a story told as an ordered sequence of events or activities. IDEF3 is a scenario-driven process flow modeling method created specifically for these types of descriptive activities. IDEF3 is based upon the concept of direct capture of descriptions of the precedence and causality relations between situations and events in a form that is natural to domain experts in an environment. The goal of IDEF3 is to provide a structured method for expressing the domain expert's knowledge about how a particular system or organization works. An IDEF3 description can be used to provide the data for many purposes including the following.

1. To provide a systematic method for recording and analyzing the raw data that results from fact-finding interviews in a systems analysis project.

2. To determine the impact of an organization's information resource on the major operating scenarios of an enterprise.

3. To provide a mechanism for documenting the decision procedures affecting the states and life-cycle of critical shared data (particularly manufacturing, engineering, maintenance, and product definition data).

4. To define data configuration management and change control policy definition.

5. To support system design and design tradeoff analysis.

6. To provide powerful mechanisms to support the generation of simulation models.

7. To provide useful information for the creation of functional (IDEFØ) models.

8. To facilitate process mapping for the design of software to achieve real-time control by providing a mechanism for clearly defining the facts, decision points, and job classifications.

9. To provide an analyst with a method to clearly define the data needed to develop needs and requirements from a user viewpoint.

10. To collect and express the views of domain experts required for the development of expert systems.

The IDEF3 Process Description Capture Method is used by system developers to capture domain expert knowledge about the "behavioral" aspects of an existing or proposed system. Process knowledge captured using IDEF3 is structured within the context of a scenario,

making IDEF3 an intuitive knowledge acquisition device for describing a system. Unlike IDEFØ models which adopt a single perspective of the system and explicitly remove all temporal logic to promote generality and simplification, IDEF3 serves to structure different user descriptions of the temporal precedence and causality relationships associated with enterprise processes. The resulting IDEF3 descriptions provide a structured knowledge base from which analysis and design models can be constructed.

The development of IDEF3 was motivated by the need to distinguish between descriptions of what a system is supposed to do and mathematical idealizations, or models, that predict what a system will do. The IDEF2 Simulation Modeling Method and a host of other simulation languages (e.g., QGERT, SLAM, etc.) are targeted at satisfying the latter concern. Such languages represent the time-varying behavior of system resources and provide a framework for the specification of math model based simulations. IDEF3 addresses the former concern as a language for the organization and expression of different user views of the system. The organizational principles and concepts upon which IDEF3 is based come from pioneering work by 1) Dr. Shir Nijssen and 2) Dr. Jon Barwise [Barwise 83, Devlin 89]. Dr. Nijssen promoted the notion that an information system is the embodiment of a discourse situation between agents in an organization. Dr. Barwise initiated an entirely new field of situation theory and situation semantics that promises to provide the theoretic basis for a new understanding of how any such discourse situation can come about and what flow of information can be supported by such a discourse situation. The interested reader is referred to [Mayer 89a, Menzel 90] for additional details regarding the theoretical background of this method.

Describing Systems From The IDEF3 Perspective

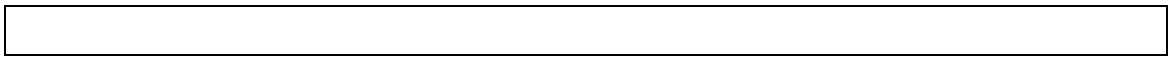
Two modeling modes exist within IDEF3: process flow description and object state transition description. Process flow descriptions are intended to capture knowledge of “how things work” in an organization. The object state transition description summarizes the allowable transitions an object may undergo throughout a particular process. Both the Process Flow Description and Object State Transition Description contain units of information that make up the description. These model entities form the basic units of an IDEF3 description. The resulting diagrams and text comprise a “description” as opposed to other IDEF methods that produce a “model.” This distinction is an important one.

As discussed previously, a model is really an idealized system of objects, properties, and relations designed to imitate important aspects of a given real-world system. In a very real sense, models are themselves systems built around assumptions and simplifications of the real-world system presumed to hold true over the range of design situations to which the model will be applied to predict real-world behavior. A model must therefore be complete and internally consistent to ensure its usefulness.

Descriptions, however, are generally incomplete. For example, one might know something about a process or event outside of his specific area of expertise, but not know everything. Or facts could be omitted from a given description as irrelevant, or simply forgotten. Descriptions are simply recordings of facts and beliefs about the world around us that are assumed to be true, but incomplete. Model construction must therefore be preceded by the accumulation of descriptions provided by domain experts.

IDEF3 Basic Concepts

An IDEF3 Process Flow Description captures a network of relations between actions within the context of a specific scenario. The intent of this description is to show how things work in a particular organization in the context of a particular problem solving (or recurring) situation. IDEF3 uses the "scenario" as the basic organizing structure for establishing the focus and boundary conditions for the process description. This feature of the method is motivated by the fact that humans tend to describe what they know in terms of an ordered sequence of activities they have experienced or observed within the context of a given scenario or situation. This natural tendency towards organizing thoughts and expressions within the context of a process description has motivated widespread use of the scenario as an informal framework for proposing alternative "external views" of possible system designs whose role will be to support the activities of the organization within the established context. Such development approaches have been referred to as "External Constraint Driven Design" approaches, and have been repeatedly demonstrated in practice as an effective mechanism for the design of new systems. Figure A-13 presents an example IDEF3 Process Flow Diagram.



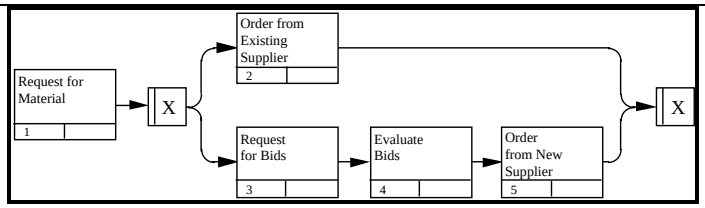


Figure A-13. An Example Process Flow Diagram

An IDEF3 Process Flow Diagram consists of the following structures:

1. Units of Behavior (UOBs)
2. Junctions
3. Links
4. Referents
5. Elaborations

The development of an IDEF3 Process Flow Diagram will consist of the generation and manipulation of these descriptive entities.

The basic syntactic unit of IDEF3 graphical descriptions used within the context of a given scenario is the Unit of Behavior (UOB). This is the name given to what may be further classified as a function, activity, action, act, process, operation, event, scenario, decision, or procedure depending on its surrounding structure. Each UOB represents a specific view of the world in terms of a perceived state of affairs or state of change relative to the given scenario. Each UOB can have associated with it both "descriptions in terms of other UOBs," otherwise called decompositions, and a "description in terms of a set of participating objects and their relations," called elaborations. Note from Figure A-13 that IDEFØ activities can be reused (and cross referenced with) IDEF3 UOBs.

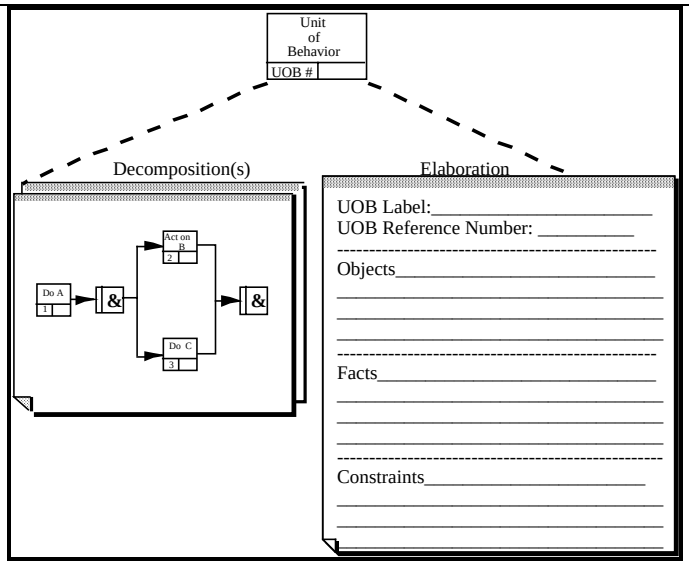


Figure A-14. UOB with Its Decomposition and Elaboration

UOBs are connected to one another through the use of junctions and links. Junctions provide the semantic facilities for expressing synchronous and asynchronous behavior among a network of UOBs. Links are 1) temporal precedence, 2) object flow, and 3) relational. Relational links are provided to permit constraint capture not accommodated by the default semantics of the precedence and object flow links.

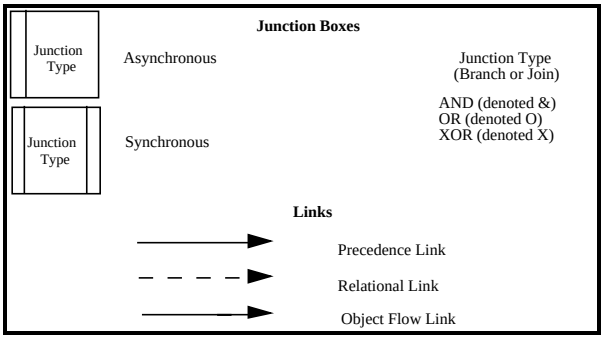


Figure A-15. IDEF3 Junction Types, Link Types, and Their Meanings

An Object State Transition Diagram is used to capture an object-centered view of a process. This view cuts across the process diagrams and summarizes the allowable transitions of an object in the domain. The entities of an Object State Transition Description are the following.

1. Object States

2. State Transition Arcs

An object state is defined in terms of property values and constraints. The properties that are kept track of by the information systems must be defined as attributes in an IDEF1 model and cross referenced in the object state Transition Diagram. An Object State can also have Pre-transition and Post-transition constraints associated with it. These constraints specify the conditions that must be met: 1) before a transition can begin, or 2) before a transition can be considered complete. The constraints are specified by a simple list of property/value pairs or by a constraint statement. The values of the attributes must match the specified values for the requirements to be met. The object state diagram also allow one to specify that an object must be processed through a particular process flow network before the transition from one state to the next is allowed. Figure A-16 shows how an object state description would appear in an Object State Transition Network (OSTN) diagram. The solid circle represents the description of the actual state. Each object state has an associated elaboration. An OSD form is constructed for every object state represented in the OSTN diagram. In addition to enabling a detailed characterization of a state, the OSD form facilitates the specification of the requirements for all possible transitions in and out of the state as well as the requirements for the object to exist in a state. There are three types of requirements which are necessary to define a state. These are 1) *entry conditions* (for an object state) that must exist for the object to transition into a state; 2) *exit conditions* (for an object state) that must exist for the object to transition out of a state, and 3) *state description* that exist while an object is in a state. These conditions are expressed as attribute-value pairs and/or constraints.

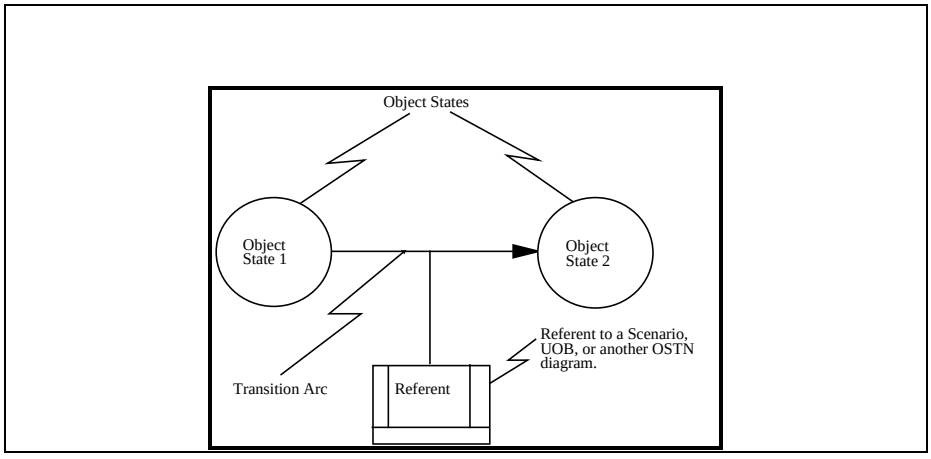


Figure A-16. IDEF3 Object State Description

Guidelines When Using IDEF3

IDEF3 is designed to provide a medium for capturing the raw description of facts known by domain experts about how their system works. However, it does provide a rich variety of mechanisms for organizing and conveying those facts. Overly complex IDEF3 diagrams can result from combining many scenarios and viewpoints into a single diagram. A heuristic to constructing IDEF3 diagrams is if any of the displayed UOBs are not visible from all of the views associated with a scenario, it should probably be in a separate UOB. Also, beware of the tendency to "fill in the gaps" in the data collected. As a description capture method, IDEF3 was designed to be tolerant of "partial" and even inconsistent descriptions. In the analysis of organizations and the information systems that support those organizations, it is often just those areas of inconsistency or incompleteness that are the root of a particular problem. If the description hides these areas either by omission or by intention, the problems may go unnoticed giving rise to the often heard comment, "These models don't indicate anything is wrong." Of course, they don't, since the modeler has polished over all of the rough areas which are exactly the areas of interest!

Object-oriented Design with IDEF4

Like IDEF1X, IDEF4 (Object-oriented Design Method) is intended to be used as a design method for automated systems implementation. IDEF4, however, targets the use of object-oriented technology, rather than relational technology, for the target implementation. The emergence of object-oriented philosophy and development practice have demonstrated the ability to produce code that exhibits desirable life-cycle qualities such as modularity, maintainability, and reusability. Likewise, the object-oriented programming paradigm has demonstrated major advancements in the ease with which software code can be created, enabling more people to successfully produce code. Paradoxically, this ease with which code can be produced also makes it easier to create software that is of poor design. Poor designs result in systems that are not modular, are difficult to maintain, and whose implementations are far more difficult to reuse. The goal of IDEF4 [Edwards 89] is to assist in the correct application of the Object-oriented Programming (OOP) paradigm to ensure consistent benefits from that technology. Before describing IDEF4, it is useful to review the background and philosophy of OOP.

Amid rising costs of serving the information needs of modern manufacturing corporations, one of the emerging technologies is OOP. Whether measured by cost, headcount, weight, or

volume, information is a major product of every world-class manufacturing corporation. The need for reductions in information production costs has led to intense interest in a technology that started for user interfaces to CAD systems in the artificial intelligence (AI) labs at the Massachusetts Institute of Technology, was used by simulation groups for many years, again became the purview of the AI community as a possible knowledge representation paradigm, and emerged as a major software productivity paradigm. The impact and benefits of this paradigm in the engineering and manufacturing domain will depend upon how well it is actually understood.

One of the current problems plaguing OOX is the nature of the X. In the scramble to acquire the technology and the mad dash to supply technology which vendors don't have, there has been the tendency to redefine object-oriented (reminiscent of the *relational*, *three schema*, *rule based*, and other buzzwords of the not so distant past). Thus, there are the terms *object-based* and *object-centered* as well as object-oriented sorts. Even Ada and IMS are presented as being "inherently" object-oriented, which, of course, might cause even the mildest skeptic to wonder what in the world is going on!

Presumably the intended goals of object-orientation was attaining goals of reusability, modifiability, reliability, maintainability, reduction of redundant code, and increased human ability to comprehend and implement more complex systems. An important feature of object-orientation in a programming language is its higher levels of abstraction. It is worthwhile to recount how the abstractions contribute to these goals.

Reusability is attained by two principal means: 1) through the encapsulation of the sets of routines operating on a data type (object class type) as a software module constituting the definition of that data type; and 2) through the concept of protocols, or declarative specifications of the purpose and effects of generic routines and their intended uses.

Modifiability and maintainability is enhanced through the same means as reusability. In addition, inheritance will make it possible to make alterations in a program simply by adding inheritance links, in cases where such alterations would require extensive rewriting of code (or design) in non-object-oriented programming languages (or design methods).

Reduction of redundant code is achieved again through the use of inheritance, which provides the effect of extensive copying of code (as in a macro definition) without any actual copying. Reducing redundancy also leads to increased reliability; redundant copies of code can easily lead to inconsistencies in much the same way redundancies in a database can.

Object-orientation is a conceptual power tool because it is so closely linked to taxonomic reasoning, which is fundamental to the human way of thinking about the ontology of problem domains. Though it must be emphasized that an object-class hierarchy is not the same as a taxonomy, the two are similar enough for object-orientation to significantly reduce the division between problem conceptualization and program coding.

The term *object* in the original computer language vernacular refers to a programming concept that supports creating and manipulating program objects consisting of some local state data (which represents relevant local state information) and some programmed behavior. A key phrase is *relevant local state information*. If one examines a large body of actual object-oriented programs, typically the classes of objects represent some information concept (possibly about a real-world object but generally not). Thus, the term *object* suffers from the phenomenon of use of a common term in a specialized manner. That is, the use of *object* in *object-oriented* in general does not refer to the physical object with the same name. Nor do object-class hierarchies in object-oriented designs normally correspond to the taxonomy hierarchy associated with the physical objects with the same names. Thus, *object-oriented programming* objects normally correspond to data things which may or may not correspond to or behave like the real thing! However, there are emerging sister paradigms to the OOP paradigm which attempt to move the conceptual power tool characteristics of the OOP up into the design, requirements, and even problem analysis stages of system development.

The following definitions are of some assistance in understanding the levels of object-orientation technology:

1. Object-based technology provides methodological analysis and conceptualization support based on analogy to real-world physical or conceptual objects. Object-based technology runs on the human brain.

2. Object-centered technology provides programming objects with local state storage, protocol, and method definition. However, the programmer must usually build his own basic inheritance structures for the object state and protocol methods.

3. Object-oriented technology provides full language and machine support (including type checking and multiple inheritance support) for the object state storage, protocol, and method definition and manipulation.

4. Persistent object-oriented technology extends the half-life of the objects providing disk resident support for storage of object-oriented structure definitions and instances.

Luckily, hardware and operating systems are advancing to the state where programmers no longer "need" to be concerned with low level details such as memory allocation. OOP based systems can raise the level of abstraction to the point where applications programmers can express the solution to problems in a clear maintainable fashion while letting the systems programmers worry about the details. This should allow widespread access to the benefits described above. As more programmers are allowed to adopt the object-oriented languages, the emerging technology of object-oriented databases (OODB) can be used to provide persistent objects, and to allow access to objects by other systems over networks. It is postulated that because of the more visible reference semantics of the OOP paradigm (i.e., it is more obvious what the data represents and how it is going to be used), system performance can improve 100 to 500 percent with OODB over relational systems.

Designing Systems From The IDEF4 Perspective

Whereas traditional methodologies, such as structure charts, data flow diagrams, etc., and data design models (hierarchical, relational, and network) have supported the development philosophy and practice for conventional systems development, IDEF4 seeks to provide the necessary facilities to support the object-oriented design decision making process. Specifically, the two primary design goals of IDEF4 are to 1) provide support for creating object-oriented designs whose implementations will exhibit desirable life-cycle qualities and reduce total implementation development time, and 2) make it easy to evaluate object-oriented code to determine whether or not the delivered product both conforms to the design and exhibits the desired life-cycle qualities.

Just as Structured Design (e.g., Structure Charts, Data Flow Diagrams) facilitated higher quality and productivity in the era of functional programming, the IDEF4 method is targeted at achieving similar advancement for object-oriented programming. Whereas the functional emphasis of structured design is paralleled by the functional emphasis of programming languages like Pascal, C, Modula-2, and Ada, the object-oriented emphasis of IDEF4 design is intended to serve the object-oriented emphasis of languages such as Smalltalk, CLOS, C++, and Eiffel. Thus, IDEF4 is intended as a mechanism for the *Design System* activity (See Figure A-2).

IDEF4 maintains information about an object-oriented design in a manner which will preserve as many of the concept and notational advances made in previous efforts. Such consistency with the previous work should assist the practicing software engineer in learning

and using effectively the IDEF4 modeling techniques. However, IDEF4 attempts to encapsulate the best practice experience of designing for object-oriented databases and programming environments. As such, IDEF4 is focused on the identification, manipulation, display and analysis of the following.

1. Object definitions including instance variables, class variables, temporary variables, and data/object types of the above variable values.

2. Object structures including the inheritance hierarchy or lattice structure and an individual objects visibility relative to other objects.

3. Individual object behavior including instantiation methods and constraints (pre-, post-, and during conditions), deletion methods and constraints, and an abstract description of the behavior

4. The protocol of the system of methods including location of the methods in the object inheritance lattice, type, number and ordering of the arguments, abstract description of the behavior of each protocol item, and specialization of the abstract behavior for the individual object classes.

As a design methodology IDEF4 was structured to expose the components of an object-oriented system design which are important for a design team to manage during the design phase of a system development process. The primary elements of an object-oriented design is the state maintenance (defined in the class structure) and the behavior sharing (described in the methods and inheritance structure). On the other hand, since IDEF4 is used to design for implementation, there may be times when a record, structure, list, string, or some other common data structure is better suited than an object. If a common data structure is used, then IDEF4 provides a means by which that type can be specified along with the class and method specifications.

IDEF4 employs a unique organization structure to ensure that design models do not become cumbersome and difficult to use with increasingly larger projects. This is accomplished by dividing the IDEF4 model into a variety of submodels, diagrams, and data sheets. In other words, IDEF4 divides the object-oriented design activity into discrete, manageable chunks, with each subactivity supported by a graphical method highlighting the design decisions that must be made along with their impact on other perspectives of the design. No single diagram shows all the information contained in the IDEF4 design model. This limits complexity and allows rapid inspection for the desired information. Carefully designed overlap among diagram types ensures consistency among the different submodels. IDEF4 is a graphically oriented methodology for the object-oriented design of computer systems. It provides the

necessary facilities to support the object-oriented design decision making process. Conceptually, an IDEF4 design model consists of two submodels: the Class Submodel and the Method Submodel. These two submodels are linked through the Dispatch Mapping, and combined capture all the information represented in a design model. However, due to the size of the Class and Method Submodels, the designer never sees these structures. Instead, the designer makes use of a collection of smaller diagrams that effectively capture the information represented in the Class and Method Submodels.

There are five diagram types used within IDEF4 to express the structure of the data object classes, inheritance relations, methods, and protocol of an object-oriented design. The following is a list of these diagrams and a brief description of their purpose.

1. Inheritance Diagrams specify the inheritance relations among classes.

2. Type Diagrams specify relations among classes defined through attributes of one class having instances of another class as values.

3. Protocol Diagrams specify the protocol for method invocation.

4. Method Taxonomy Diagrams classify method types by behavior similarity and links between class features and method types.

5. Client Diagrams illustrate *clients* and *suppliers* of methods.

There are also two specialized Data Sheets that accompany the diagrams.

1. Class Invariant Data Sheets (specify constraints which apply to every instance of a particular class of objects).

2. Contract Data Sheets (specify contracts that the methods in a method set must satisfy).

Understanding these diagrams requires a description of the basic concepts of IDEF4. These concepts are presented in the next section.

IDEF4 Basic Concepts

The concepts that exist within IDEF4 will be familiar to those with object-oriented experience. The same structures that exist in most object-oriented languages also exist in IDEF4. The most notable concepts in IDEF4 are:

1. Classes

2. Features

3. Inheritance Links

4. Type Links

5. Method Sets

The class is the basic syntactic unit in an IDEF4 design model. The characteristics of a class are represented by a collection of features. Each feature can be either public or private, where a public feature is accessible to all classes and a private feature is accessible only by the class and its subclasses. The power of the object-oriented paradigm comes through the inheritance of classes. When an inheritance relationship is specified, all features of the parent class (superclass) are passed on to the child class (subclass). When this occurs, the inherited features in the subclass maintain the same characteristics as in the superclass unless they are explicitly redefined. This inheritance provides the ability to build complex class structures from simple classes. Figure A-17 shows an example class inheritance diagram and the effect of both single and multiple inheritance on a simple set of graphics objects.

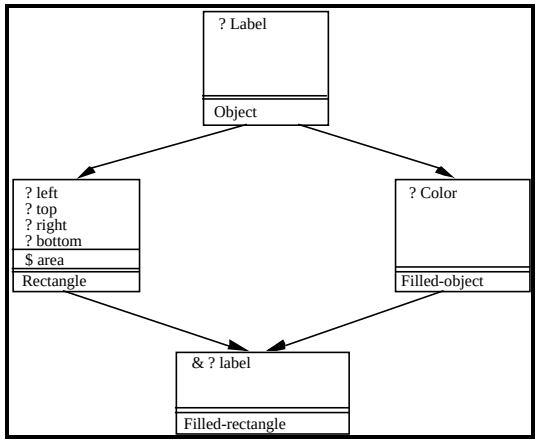


Figure A-17. A Class Inheritance Diagram in IDEF4

Figure A-18 shows how a class would appear in an IDEF4 Class Inheritance Diagram. The class is represented by a square-cornered box with the name of the class listed below the double line at the bottom of the box. IDEF4 requires that the first letter of the class name be capitalized. The features of the class are also displayed in the class box with private features displayed below the export line and public features displayed above the export line. The

feature symbols provide additional information about the role that the particular feature plays.

For each class defined, the designer will attach a Class Invariant Data Sheet. This sheet is used to provide additional information about the objects in a class. The information represented in this data sheet must be true for all objects in the class at all times. An interesting feature of the Class Invariant Data Sheet is that subclasses also inherit the Data Sheet constraints of their superior.

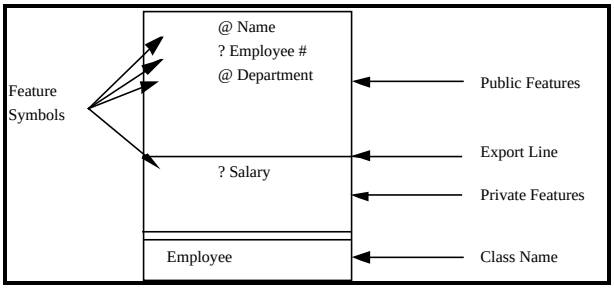


Figure A-18. A Class Box in IDEF4

A feature is the named representation of a particular characteristic of a class. The features are used to capture the behavior of instances of a particular class. When the designer defines a feature, the type of feature must be specified. It is important to distinguish between the type of feature and the type of value of the feature. Feature value type is concerned with the legal values that a feature may take or return. Feature type is concerned with the role that a feature will play within the context of a class. A feature can be only one of six types in an IDEF4 design model:

1. Non-Specific Feature (used as a place-holder early in the design process)
2. Routine (used to specify a feature that is to be implemented as a program)
3. Attribute (used to specify a feature that will hold or calculate a value)
4. Function (feature with characteristics of both a routine and an attribute)
5. Procedure (routine that is executed only for its side effect, doesn't return a value)
6. Slot (attribute that "holds" a value)

An Inheritance Link simply defines a parent/child relationship between two classes. When an inheritance link is specified, all characteristics of the parent class are inherited in the child class. Also, the inherited features will exhibit the same behavior in the child class as in the parent class unless the features are redefined using the auxiliary feature symbols.

A class can be considered to be a data type, and traditional programming data types can be considered to be classes. Since the features of classes that are classified as Attributes, or more specifically Slots or Routines, take on values, it would be useful to indicate the type of value that the feature will take. These type declarations are made with Type Links. The Type Link specifies the feature being typed and the class that represents the legal values that the Attribute may take.

Figure A-19 shows the four different Type Links supported in IDEF4. The first link simply says that the Attribute f of A returns a value of Type B. The second link is identical to the first except there is also a dual: while f of A returns a value of Type B, g of B returns a value of Type A. This dual link reduces the number of links that might appear in a Type Diagram and thus provides for a simpler diagram. The third link indicates that f of A returns a value that is constructed from B. This may be a list of instances of B or some other structure composed of instances of B. Finally, the fourth link provides a semi-dual for the third link type. While f of A returns a value constructed from instances of B, g of B returns a value of Type A. Figure A-20 shows a type diagram for the graphical object example.

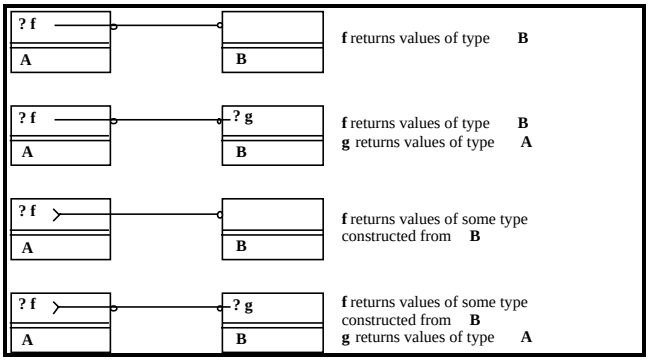


Figure A-19. IDEF4 Link Types

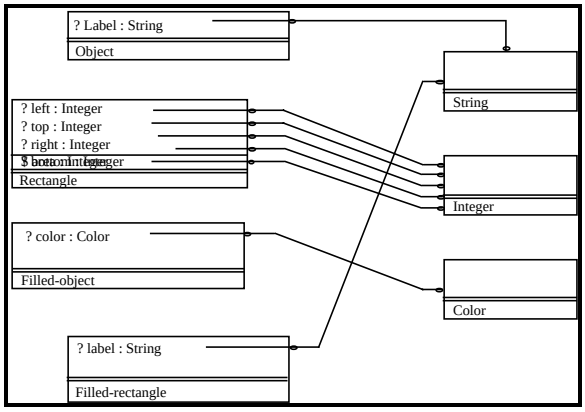


Figure A-20. IDEF4 Type Diagram

IDEF4 does not represent individual methods. The reason for this is that a method could accept parameters that are instances of more than one class. As a result, the same method must be defined for both classes. To alleviate this repetition and confusion, the information of these methods will be maintained in a method set. A feature of a class and that class will map to a method set. This mapping is referred to as Dispatch Mapping.

A method set is defined by its associated contract. In actuality, a method set is just a name for a contract data sheet. The contract data sheet maintains the declarative statements that define the intended effect of the methods in the method set. For a function, the contract would state the relationship between the function's argument list and the corresponding

return values. For a procedure, the contract would have to specify how the method set changed the environment when passed an argument list and the current environment.

Guidelines When Using IDEF4

Many users of OOP have adopted the technology under the assumption that it eliminates the need for requirements analysis and design. This results in the same types of cultural problems with the use of IDEF4 as were experienced in the early days of traditional structured design (e.g., experienced OOP programmers don't see the reason, novice don't understand the concepts, and managers were hoping to eliminate a step in the life-cycle process.) Besides these cultural problems the IDEF4 method provides means for assisting in the focus on behavioral abstractions. However, new users typically use taxonomic abstractions in their formulation of the type diagrams. The natural consequence of this is that the method mapping between the type diagrams and the method set diagrams becomes very complex. This complexity is IDEF4's way of indicating to the designer that the class structure needs revision. Finally, IDEF4 was designed to assist in the design process. The assumption of IDEF4 was that it would be supporting an underactive design decision making process. However, it is quite common for the user of IDEF4 to treat the method simply as a means of documenting a design, often one that has already been coded.

Concept / Ontology Descriptions With IDEF5

In the context of information management, *ontology* is the task of extracting the nature and structure of a given engineering, manufacturing, business, or logistical domain and storing it in an usable representational medium. There is a pressing need for methods that can effectively capture what is known about the real-world and the relationships that exist between people, places, events, etc. The importance of capturing ontological information is especially crucial in the context of large systems. A large CIM project involves coordination of the resources of many different clusters of cooperative organizations. Each cluster makes its own contributions, and the overall success of the project depends on the degree of integration between those different clusters throughout the development process. A key to effective integration is a system ontology that can be accessed and modified across clusters that captures common features of the overall system relevant to the goals of disparate clusters. This *common* framework 1) promotes sharing of information arising from various sources within the system, 2) eases problems of information base maintenance, and 3) enhances the reusability of information once collected.

Rapid acquisition of reliable systems is perhaps the strongest motivation for ontology. Among the most significant problems in information management is the redundant effort expended capturing or re-creating information in systems that has already been recorded and developed. Rarely is this redundant effort expended purposefully. Rather, it is often the result of the inability of the development team to recognize the similarities or equivalencies between the situations. IDEF5 is targeted at the construction of reference models that can be used as a basis for both manual and automated identification of these similarities. IDEF5 can also be used as a precursor to enterprise information analysis. It provides a structure for recording and organizing the raw knowledge about physical and conceptual objects along with their natural associations. This fact base can then be used as the basis for a variety of analysis purposes (including determination of the information implications of these concepts). Finally, ontological analysis has been demonstrated to be an effective first step in the construction of robust knowledge based systems [Hobbs 87]. The current generation of CIM implementations will be taking advantage of knowledge based and expert systems technology. IDEF5 provides a method for the initial knowledge acquisition for these systems. It also provides a representation of that knowledge that is independent of any particular implementation shell.

Ontologic inquiry has been the subject of extensive work within the information sciences. IDEF5 developments have drawn from such foundation work as Semantic Nets, Situation Theory, the NIAM Object Role Model, IDEF1, set theory, first-order predicate logic, modal logic, etc. In doing so, IDEF5 attempts to fill a methodological gap not targeted by any other existing methodology. IDEF1 and IDEF1X capture primarily structural information. IDEFØ and IDEF3 capture various types of process information. Of course, since both structural information and process information involve objects in a system, there is the capacity for limited ontology representation within the existing methodologies. But, as noted below, there are several important kinds of ontological information that are not representable in those methodologies. Furthermore, those methodologies do not include techniques specifically designed for eliciting and capturing system ontologies. This suggests that there is a need for a separate method.

IDEF5 models the concepts and the conceptual relations for a domain. Conceptual modeling provides an abstract level of representation for describing a problem domain and/or system which closely reflects the human conceptualization of that domain including representation of real-world objects, attributes, and functions. An ontology represents the theory of what exists in a domain.

Describing Systems from the IDEF5 Perspective

An ontology can be thought of as a structure for representing knowledge about the world as perceived from different perspectives such that those perspectives can be related to one another. Ontologies are concerned with the identification and classification of concepts, objects, and associations together with the essential characteristics that identify the those kinds and associations.

Defining an ontology for a domain involves four major activities [Menzel 91]: 1) providing an inventory of the kinds of objects that exist within a given domain according to best sources of information regarding that domain (e.g., a domain expert); 2) for each kind of object, providing a description of the properties that are common to all and only instances of that kind; 3) characterizing the particular objects that in fact instantiate the kinds within a system; and 4) providing an inventory of the associations that exist within a given domain between (and within) kinds of objects.

For example, consider the semiconductor manufacturing industry. The first two tasks might identify kinds of objects including wafers and reagents. Reagents may represent several

subkinds including liquid reagents and etchants. Each of these kinds would have an associated set of necessary properties that its members contain.

The third and fourth tasks of ontology become more relevant in contexts where we want to be able to characterize specific individual objects, to speak specifically of them, their properties, and their associations. A basic distinction that is incorporated into IDEF5 is the distinction between essential and accidental properties. An essential property of an object is simply a property that the object could not possibly have lacked. An accidental property, by contrast, is a property that an object in fact has, but nonetheless might not have. The following section discusses how this feature in IDEF5 supports the rapid development of usable ontologies in the manufacturing domain which must deal with a rich combination of both natural and human designed objects.

IDEF5 Basic Concepts

The notion of "kind" (as distinct from class or type) is a central concept for IDEF5. It is important to recognize the distinction between the usual meaning of "kind" and what it represents in IDEF5. In naturally occurring systems it is often the case that all objects of the same kind have a distinguishing set of properties that must be maintained to remain a member of that kind. That is to say that the properties for membership are essential properties of the member. Thus, the usual notion of a kind is that of a collection of objects, all of which share a common nature, i.e., a set of properties that belong essentially to all and only members of the kind. However, in the manufacturing systems it is frequently the case that objects must have a certain set of properties to become part of a kind but are not required to keep those properties to remain part of the kind. Consider semiconductor manufacturing as discussed above. A chemical has certain properties that identify it as an etchant, and all etchants have those properties. This is the traditional idea of a natural kind. Contrast this with the kind of object that a manufacturing "rework" item represents. A rework item might be any wafer that has more than three defects. Therefore, a wafer with four defects becomes a rework item. However, after one or more of the defects on a wafer is repaired, it is still a rework item. In fact, it remains a rework item until it is reclassified by an inspector as an acceptable wafer or is discarded. This is an example of the "kinds" that typically arise in human designed systems. IDEF5 supports the identification of both kinds of kinds.

Put another way, the reason for the broader notion of a kind is that when an ontology is built for a certain human designed system, we are not just setting out to discover and classify the world as it is in itself, but rather to divide up and categorize the objects within the system in useful and informative ways. An ontology's categorization scheme is justified only insofar as it is useful to organizing, managing, and representing knowledge or information in the system so categorized. If objects of a certain kind K play a useful role in the system, that is all the justification one needs for admitting them into the system's ontology, irrespective of whether or not the defining properties of K are essential to its members (ask yourself, Is it necessary that your manager know how to manage?).

There is more to characterizing the objects in a system than listing their properties, though. For in the context of a given system it is equally important to detail the associations that objects in the system can, and do, bear to one another. Just as with properties, system-essential associations must be distinguished from system-accidental associations. This is partially because associations occur that way. It is also because the association may be a defining property of a kind (e.g., the marriage association and the kind "married"). A system-essential association relative to two (or more) kinds K_1, K_2 is an association that must hold whenever there are instances of K_1 and K_2. A system-accidental association relative to K_1 and K_2, by contrast, is one that need not hold between all possible instances of those kinds. Note that, just as defining properties of kinds neednot be essential to their instances, in the same way objects that stand in system-essential relations don't necessarily stand in those relations essentially; in human designed systems.

IDEF5 provides three diagram types for the visualization of an ontology. These diagrams are useful in both the construction and validation of the ontology.

Is-a diagrams are used in IDEF5 to show "is-a" relationships between kinds in an IDEF5 model. IDEF5 provides three types of is-a links: 1) generalization/specialization, 2) AKO (a kind of), and 3) description subsumption. These link types are taken from [Brachman 83].

Generalization/specialization links (also called superset/subset links) represent the specialization of a kind by another kind. For example, a **hex-headed bolt** kind is a specialization of a **fastener** kind for bolts with hex heads. AKO links are useful for representing natural kinds. For example, a **dog** kind is a kind of a **mammal** kind. Description subsumption links are useful for representing abstract kinds. The fact that "a square is a rectangle with four equal sides" is captured in a description subsumption link.

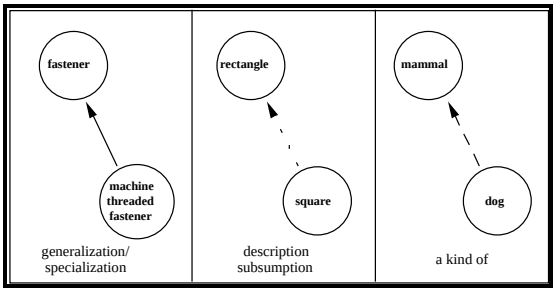


Figure A-21. Is-a Link Types

System kind diagrams are used in IDEF5 to show the kinds and relations that make up a system. Figure A-22 is an example of a system kind diagram for a "Wafer Cutting System." Kinds in the system are represented by circles. System kinds in the system are represented by double circles. Lines between the circles represent relations, with the relation being from the tail of the arrow to the head of the arrow. System-essential relations are shown by double lines, and system-accidental relations are shown by single lines.

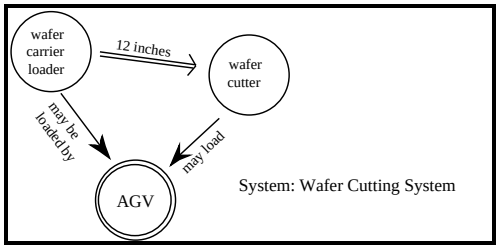


Figure A-22. System Kind Diagram

The third type of diagram in IDEF5 is the *relation type diagram*. This diagram shows the *axiomatization* of a relation in the model. To axiomatize a relation is to describe its meaning in terms of other relations in the system. This has traditionally been one of the most difficult parts of ontology development. The idea behind the relation type diagrams is to maximize reuse of a core set of relation definitions. Figure A-23 shows a relation type diagram for the **has_sealant** relation.

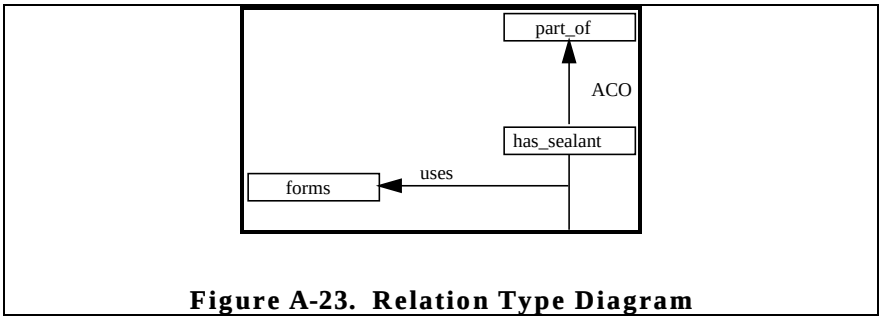


Figure A-23. Relation Type Diagram

The **has_sealant** relation occurs between a **fastener** kind and a **sealant** kind. This relation reflects the fact that, in the automotive domain, there are fasteners that require sealants for some applications. The two relation type specifiers, as shown in the figure, are *ACO* (abides by the contract of) and *uses*. ACO links allow the ontology developer to reuse axioms that characterize one relation in the description of another. The uses link allows quick determination of the interrelationships between the relation characterizations.

Besides the diagram types IDEF5 includes a ten step process for the construction and application of the method. A more complete discussion of the theoretical foundations for the method can be found in [Menzel 91]. An in-depth description of the actual method can be found in [KBSI 91].

Things To Consider When Using IDEF5

IDEF5 provides the basic concepts for organization of the process of ontology development in a domain. Initial experience with this process indicates that it involves an iterative set of activities of: 1) fact data collection and analysis; 2) discovery of "proto-kinds" (initial guesses at the kinds); 3) refinement of the proto-kinds and their associations; 4) validation of the ontology against the original facts. One of the lessons learned is that steps 2 and 3 appear to be best performed by single individuals working alone on separate chunks of the data, where as steps 1 and 4 can easily be performed by a team of modelers.

Information System Design Rationale Capture With IDEF6

Advancement in technology, manufacturing methods, and materials has brought about the emergence of products whose expected usable lifetimes extend over decades and even centuries. Information systems too, have evolved from stand alone application oriented

systems with relatively short lifetimes and limited scope towards large scale, distributed systems which must service their users over extended periods of time. Not unlike traditional products, maintenance of information systems whose expected lifetimes may extend over many career periods required explicit capture and storage of the rationale used in their design.

When explicitly captured, design rationale typically exists in the form of unstructured textual comments. In addition to making it difficult, if not impossible to find relevant information on demand, lack of a structured method for organizing and providing completeness criteria for design rationale capture make it unlikely that important information will be documented.

Unlike design methods (like IDEF1X, IDEF2, and IDEF4) which serve to document WHAT a design is, new methods are needed to capture WHY a design is the way it is, or WHY it is not manifested in some other form, together with HOW the final design configuration was reached. For the purpose of this discussion, *Design Specification* means capture of WHAT a design is; *Design Rationale* indicates WHY, WHY NOT, and HOW a design arrived at its final configuration; and *Design History* indicates the time-ordered sequence of steps used in the realization of the design.

IDEF6 is intended to be a method with the representational capability to capture information system design rationale and associate that rationale with the design models and documentation for the end system. Thus, IDEF6 attempts to capture the logic underlying the decisions contributing to, or resulting in, the final design. The explicit capture of design rationale serves to help avoid repeating past mistakes, provides a direct means for determining the impact of proposed design changes, forces the explicit statement of goals and assumptions, and aids in the communication of final system specifications. Explicit capture of the motivations for why a designer selected or adopted a particular design strategy or system feature for enterprise level information systems is essential to the maintenance of that system over its life-cycle.

Design Rationale from the IDEF6 Perspective

The purpose of IDEF6 is to facilitate the acquisition, representation, and manipulation of the design rationale utilized in the development of enterprise level information systems. The term 'rationale' is interpreted as the "reason, justification, underlying motivation, or excuse" that moved the designer to select or adopt a particular strategy or system feature. More

simply, 'rationale' is interpreted as the nature of the answer given to the question "Why is this design the way it is?" IDEF6 is intended to be a method with the concepts and language capabilities needed to represent information about the situations, relations, objects, states of affairs or courses of events that constitute system design rationale and associate that rationale with design specifications, models and documentation for the system. The scope of applicability of the technique component of IDEF6 is in the conceptual, preliminary and detailed design activities of the information system development process. To the extent that detailed design decisions for software systems are relegated to the coding phase the IDEF6 technique should be usable during the software construction process. Assumptions associated with the scope of IDEF6 include:

1. IDEF6 is targeted towards facilitation of the capture of design rationale for enterprise level information systems from the system level design to the detailed design of the implementation data structures, algorithms, user interface and processes.

2. It is unreasonable to expect designers to sit down at some point in time and "model" design rationale. Rationale must be captured at the source–at the point in time at which decisions are made.

3. People rarely write down design assumptions or rationale. To the extent possible it must be the case that IDEF6 be incorporated in a transparent manner into a wide variety of design methods (both formal and informal).

4. Design rationale is a small part of development decision rationale. That is, assume that design rationale will reference decisions on "what are important symptoms" and decisions defining what are the problems that give rise to those symptoms.

A general characterization of design rationale can be given as: "The beliefs and facts as well as their organization that the human uses to make design commitments and propagate those commitments." IDEF6 characterizes both "types" of design rationale and "mechanisms" for representation of these types. Types of design rationale identified for IDEF6 capture include:

1. Philosophy of a design including:

   A. Process descriptions of intended system operation.

   B. Design themes in terms of object or relation types.

2. Design limitations expressed as range restrictions on system parameters or environmental factors.

3. Factors considered in trade-off decisions.

4. Design goals expressed in terms of:

    A. Use or lack of use of particular components.

    B. Priorities on a problems requirements.

    C. Product life-cycle characteristics (e.g., disposable versus maintainable).

    D. Design rules followed in problem or solution space partitioning, test/model data interpretation, or system structuring.

5. Precedence or historical proof of viability.

6. Legislative, social, professional society, business, or personal evaluation factors or constraints.

Possibly due to the commonness of the carry-over strategy or the complexity of design rationale expression, the most common rationale given for a design is that it was the design that worked last year. Without making judgment on this situation, a minimum requirement for a design knowledge rationale capture capability is the ability to record historical precedence, as well as statements of beliefs and rationalizations for why a current design situation is identical to the one the previous design serviced. Another important rationale given for a design is just "it feels better," "it seems more balanced, symmetric." There is an important aesthetic side to software design.

Software design rationale includes expectations about how the design will evolve through the development process itself. For example, expectations about how the program structure will probably change–note such expectations do not appear to be as well defined as similar expectations we have seen in mechanical hardware design.

IDEF6 Basic Concepts

Thus, IDEF6 can be viewed as a structured technique for formulation of the types of design rationale statements (e.g., philosophy of the design, range restrictions or constraints, factors considered in trade-off decisions, design goals, etc.) It is also capable of supporting the formulation of such statements by simple reference to other life-cycle artifacts or objects. That is, if the reason for a particular design element is to satisfy a particular requirement constraint, then IDEF6 allows the statement of just this relationship with references to the requirement constraint (eliminating the need to reproduce the requirement constraint in the

IDEF6 language). IDEF6 is still in its formulative stages. At this point of time, it takes the form of a language for the following.

1. Stating rationale.
2. Associating rationale statements with design elements.
3. Making and classifying "rationale" links between design elements and other life-cycle objects.

While IDEF6 could be applied in purely manual form, it is best suited for application in an automated environment that includes a life-cycle artifact repository (e.g., the IBM AD-cycle or DEC repositories or the Air Force Integrated Development Support Environment – IDSE). The IDEF6 language is based on an ontology of design rationale. That is, the language includes (as key words) a set of commonly used terms or phrases that express elements of rationale. An example of such a term and a phrase is the term "satisfies" and the phrase "is satisfied by" used in the following structures.

1. Design feature A *satisfies* the requirement B.
2. Requirement B *is satisfied by* design feature A.

Other terms/phrases that must be considered in an ontology of design rationale would include the following.

System

Subsystem

Component

Requires/Is Required By

Constrains/Is Constrained By

Bounds/Is Bounded By

Supports/Is Supported By

Creates/ Is Created By

Translates/Is Translated By

The IDEF6 language structure provides simple structured English-like sentence forms for employing these "rationale forming" constructs into statements associated with a design of a particular CIM system.

Issues In Design Rationale Capture

Since IDEF6 is still in the formulative stages, this paper describes factors associated with its application that have been discovered in the development process. For example, it is unreasonable to expect designers to introduce a separate step in the design process to document, or model, the assumptions or rationale upon which a given design decision is based. Therefore, much of the capture of this information must occur through background processes or interactive questioning initiated by a design support environment rather than the designer. This has influenced the IDEF6 method development in that it is assumed that the method will be used simultaneous with a number of different system design methods. Another important issue that has surfaced is that design rationale is just a part of the rationale motivating a development decision in the first place. Design rationale must therefore reference the symptoms motivating the system development decision and the probable causes giving rise to those symptoms.

Closing Remarks

Referring to Figure A-2, this paper has presented some of the IDEF family of methods that have been used or are being developed to accomplish the analysis and design of a CIM system. Figure A-24 shows a list of the current IDEF methods being developed. The reader may question the number of methods and the lack of one encompassing method capable of representing all that is needed to know about an existing or proposed system. Intuitively, it would be nice to have a single method representing all relevant perspectives of the system. This question can be answered by considering the purpose of models and descriptions from a slightly different perspective.

Generally speaking, the purpose of models and descriptions is to help make decisions. Each type of model or description focuses on a relatively narrow set of relationships and system characteristics comprising a particular viewpoint or perspective of the overall system. Analysis models, for example, are used to determine existing or anticipated design requirements. Design models serve to facilitate optimization of desirable design features for a restricted set of system requirements. Simulation models provide a perspective from which various measures and statistics associated with system performance can be generated to examine specific performance characteristics under a restricted set of operational conditions. Each model and the decisions generated through its construction carries with it a relative

weighting towards overall system level decisions. Competing design decisions highlighted within and between model types eventually emerge, necessitating trade-offs.

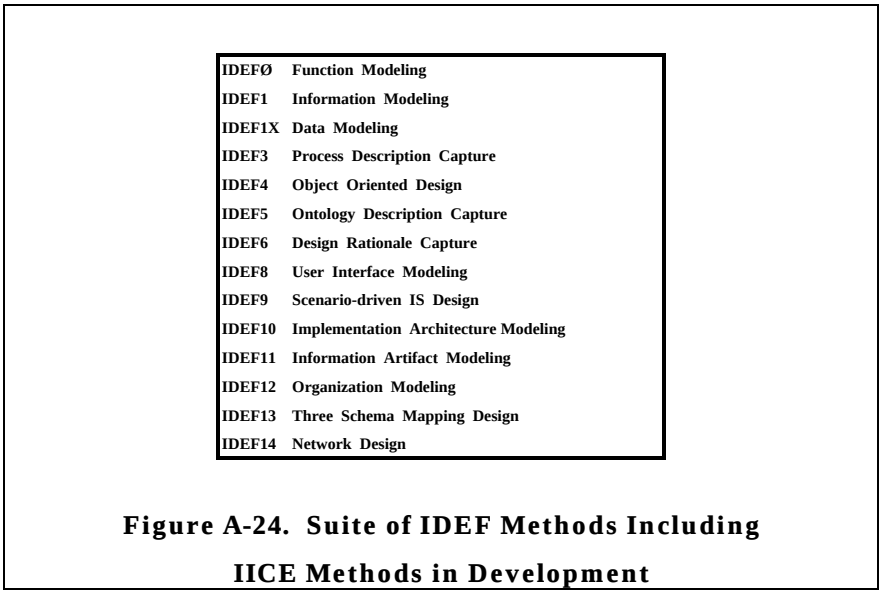


Figure A-24. Suite of IDEF Methods Including IICE Methods in Development

The goal of this process is an *optimal* design of the proposed system. Of course, designs or systems are considered optimal when evaluated against the current set of values, each of which is somehow manifested in the trade-off decisions made. This means that an optimal design does not necessarily, and most likely won't, exhibit all desirable life-cycle or performance characteristics.

As a result, models and descriptions focus on a limited set of system characteristics and explicitly ignore those characteristics not directly pertinent to the decisions at hand. Models and descriptions were never intended to represent every possible state or behavioral characteristic of a system. If such a goal were achievable, the exercise would itself constitute building the actual system, thus negating the benefits of modeling (e.g., low cost, rapid evaluation of anticipated performance, etc.). Having extended beyond the bounds of modeling into the realm of actual system construction, simulation becomes a statistical exercise rather than a design decision-making process.

The tendency to seek a single model to represent all relevant system life-cycle and behavioral characteristics, therefore, would necessitate skipping the design process altogether. Similarly, the search for a single method, or modeling language, to facilitate conceptualization, system analysis, and design continues to frustrate those making the attempt. Recognizably, the plethora of special purpose methods which typically provide few,

if any, explicit mechanisms for integration with other methods, is equally frustrating. The IDEF family of methods is intended to strike a favorable balance between special purpose methods whose effective application is limited to specific problem types, and "super methods" which attempt to include all that could ever be needed. This balance is maintained within the IDEF family of methods by providing explicit mechanisms for integrating the results of individual method application.

Perhaps the most compelling argument for a family of methods is the ever-increasing need for methods that help manage complexity by dividing up the systems that must be analyzed, designed, and developed into discrete, manageable chunks. Methods are designed to embody knowledge of good practice for a given analysis, design, or fabrication activity. An appropriately designed method serves to raise the level of performance of the novice to a level comparable with that of an expert by focusing the modeler's attention on important decisions while masking out irrelevant information and unneeded complexity.

For the customer, floor plans and artists renderings are just as important as the final blueprints. It is therefore incumbent on the methods developer to constantly re-evaluate how well individual methods serve the needs of both the modeler and the customer. Practitioners must become sufficiently familiar with the basic theory behind the methods to ensure their appropriate selection and use.

Just as the original IDEF methods were targeted at managing the complexity associated with evolution towards large-scale integration in the manufacturing environment, new challenges will continue to emerge as those visions extend to integration across traditional boundaries as well. Large-scale integration between engineering, manufacturing, and support activities will be both exciting and challenging, particularly to the methods engineer. Their task will be to encapsulate the basic theory and body of experience associated with the analysis, design, and realization of tomorrow's integrated environments in easily usable forms

Bibliography

[ANSI 75] ANSI/X3/SPARC, Study Group on Data Base Management Systems: Interim Report, 75-02-08, In: ACM SIGMOD Newsletter, FDT, Vol. 7, No. 2, 1975.

[Barwise 83] Barwise, J. and Perry, J., *Situations and Attitudes*, The MIT Press, Cambridge, 1983.

[Cullinane 90] Cullinane, T., McCollom, N., Duran, P., and Thornhill, D. "The Human Elements of IDEF," Unpublished Paper, May, 1990.

[Devlin 91] Devlin, K., *Logic and Information, Volume I: Situation Theory*, Cambridge University Press, 1991.

[GE 85] General Electric, Integrated Information Support System (IISS). Volume 5. Common Data Model Subsystem. Part 4. Information Modeling Manual. IDEF1 Extended. DTIC-A181952, December, 1985.

[Feldmann 89] Feldmann, C.G., "Levels of Abstraction in IDEFØ Models," Unpublished Paper, October 10, 1989.

[ISO 87] International Standards Organization, "Information Processing Systems – Concepts and Terminology for the Conceptual Schema and the Information Base," ISO/TR 9007, July 1, 1987.

[IUG 90] IDEF Users Group, "*IDEF – Framework, Draft Report*," IDEF-U.S.-0001, Version 1.2, Working Group 1 (Frameworks), Technical and Test Committee, IDEF – Users Group, May 22, 1990.

[KBSI 91a] Knowledge Based Systems, Inc. (KBSI). The Nature of Ontological Knowledge: A Manufacturing Systems Perspective. *KBSI Technical Report Number KBSI-SBONT-91-TR-01-1291-01*, 1991.

[KBSI 91b] Knowledge Based Systems, Inc. (KBSI). Formal Foundations for an Ontology Description Method. *KBSI Technical Report Number KBSI-SBONT-91-TR-01-1291-02*.

[KBSI 91c] Knowledge Based Systems, Inc. (KBSI). Ontology Acquisition Method Requirements Document. *KBSI Technical Report Number KBSI-SBONT-91-TR-01-1291-03*.

[KBSI 91d] Knowledge Based Systems, Inc. (KBSI). Ontology Capture Tool Requirements Document. *KBSI Technical Report Number KBSI-SBONT-91-TR-01-1291-04*.

[KBSI 91e] Knowledge Based Systems, Inc. (KBSI). ***IDEF5*** *Method Report*. Prepared for U.S. Air Force Human Resources Laboratory, Contract No. F33615-C-90-0012.

[KBSI 91f] Knowledge Based Systems, Inc. (KBSI). Reliable Object Based Architecture for Intelligent Controllers. *DARPA SBIR 91-050*. Contract No. DAAH01-91-C-R235.

[KBSI 91g] Knowledge Based Systems, Inc. (KBSI). "Knowledge Based Information Model Integration," Final Technical Report, *NSF SBIR* . Award No. ISI-9060808, 1991.

[KBSI 92a] Knowledge Based Systems, Inc. (KBSI), 1992. Ontology Capture Tool: Object-oriented Design Document. *KBSI Technical Report Number KBSI-SBONT-91-TR-01-0292-01*.

[KBSI 92b] Knowledge Based Systems, Inc. (KBSI), 1992. Knowledge-Based Automated Process Planning System with Assumption-Based Truth Maintenance System and Geometric Reasoning System. *DARPA SBIR 91-223*. Contract No. DAAH01-92-C-R066.

[KBSI 92c] Knowledge Based Systems, Inc, (KBSI), 1992. *IDEF3 Method Report*, Prepared for U.S. AL/HRG, Contract Number: F33615-90-C-0012.

[KBSI 92d] Knowledge Based Systems, Inc, (KBSI), 1992. *IDEF4 Method Report*, Prepared for U.S. AL/HRG, Contract Number: F33615-90-C-0012.

[Mayer 87] Mayer, R.J., et al., "Knowledge-Based Integrated Information Systems Development Methodologies Plan." Volume 2, DTIC-A195851, December, 1987.

[Mayer 90a] "IDEFØ Function Modeling: A Reconstruction of the Original Air Force Report," Mayer, R.J., editor, Knowledge Based Systems, Inc. College Station, TX, 1990a.

[Mayer 90b] "IDEF1 Information Modeling: A Reconstruction of the Original Air Force Report," Mayer, R.J., editor, Knowledge Based Systems, Inc. College Station, TX, 1990b.

[Mayer 90c] "IDEF1X Data Modeling: A Reconstruction of the Original Air Force Report," Mayer, R.J., editor, Knowledge Based Systems, Inc. College Station, TX, 1990c.

[Mayer 91a] Mayer, R.J., Menzel, C.P., and Mayer, P.S.D., "IDEF3: A Methodology for Process Description," Final Technical Report, Integrated Information Systems Evolution Environment Project, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, August, 1991.

[Mayer 91b] Mayer, R.J., Edwards, D. A., Decker, L. P., and Ackley, K. A., "IDEF4 Technical Report," Integrated Information Systems Evolution Environment, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, July, 1991.

[Mayer 91c] Mayer, R.J., deWitte, P., Griffith, P., Menzel, C.P., "IDEF6 Concept Report," Integrated Information Systems Evolution Environment, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, July, 1991.

[Mayer 91d] Mayer, R.J., deWitte, P., *Framework Research Report*, Final Technical Report, Integrated Information Systems Evolution Environment, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, June, 1991.

[Mayer 91e] Mayer, R.J., Painter, M., "IDEF Family of Methods," Technical Report, Knowledge Based Systems, Inc., College Station, TX, January, 1991.

[Mayer 91f] Mayer, R.J., et al., "Integrated *Development Support Environment (IDSE) Concepts and Standards*, Final Technical Report," Integrated Information Systems Evolution Environment Project, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, July, 1991.

[Mayer 91g] Mayer, R.J., Decker., L., "ISyCL Technical Report," KBSL-89-1002, Knowledge Based Systems Laboratory. AFHRL, Wright-Patterson Air Force Base, OH, 1991.

[Menzel 90] Menzel, C.P., Mayer, R.J., and Edwards, D., "IDEF3 Process Descriptions and Their Semantics," Kuziak, A., and Dagli, C., eds. *Knowledge Base Systems in Design and Manufacturing*, Chapman Publishing, 1990.

[Menzel 90] Knowledge Based Systems Laboratory. *IDEF3 Formalization Report.* Integrated Information Systems Evolution Environment, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, 1990.

[Menzel 91] Menzel, C.P., and Mayer, R.J., "IDEF5 Concept Report," Final Technical Report, Integrated Information Systems Evolution Environment, United States Air Force AL/HRGA, Wright-Patterson Air Force Base, OH, July, 1991.

[Painter 90] Painter, M.P., "Modeling with an IDEF Perspective: Some Practical Insights" Proceedings, SME Autofact 90, Detroit, MI, 1990.

[Ross 85] Ross, D.T., "SADT Today: A Retrospective On An Idea." *IEEE Computer Magazine, 1985 Special Issue on Requirements Engineering*, June, 1985.

[SEM 83] "Analysis of IDEF Method Application in Industrial Practice," Interim Technical Report, Systems Engineering Methodology Program, Hughes Aircraft Corporation.

[Soley 90] Soley, R.M. (ed.), "Object Management Architecture Guide," Object Management Group, Inc.

[Softech 81a] Softech Inc., "Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II, Volume IV, Function Modeling Manual (IDEFØ)," DTIC-B062457, June, 1981a.

[Softech 81b] Softech Inc., "Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II. Volume V. Information Modeling Manual (IDEF1)," DTIC-B062458, June, 1981b.

[Softech 81c] Softech Inc., "Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II. Volume VI. Dynamics Modeling Manual (IDEF2)," DTIC-B062458, June, 1981b.

[Williamson 90] Williamson, W.R. "Effective IDEFØ Modeling—Some Tricks of the Trade," Unpublished Report, May, 1990.

[Zachman 87] Zachman, J., "A Framework for Information Systems Architecture", *IBM Systems Journal*, Vol. 26 No. 3, September, 1987, pp. 276-292.

Appendix B

Knowledge Based Systems, Inc.

One KBSI Place
1408 University Drive East
College Station, Texas 77840-2335
(409) 260-5274

Profile

Knowledge Based Systems, Inc. (KBSI) is a Texas-based corporation with offices in College Station, Texas and Detroit, Michigan. Knowledge Based Systems, Inc. specializes in innovative software solutions and products in areas including expert systems for engineering, manufacturing and maintenance, and tools for systems/software engineering. Because of close university and industry ties, a major thrust within KBSI is the translation of research results into new products designed to address industry needs. KBSI is the Air Force contractor responsible for the continued development and maintenance of the public domain IDEF system/software definition, design, and engineering methods.

Current KBSI software products include the following:

AIØ™	An interactive PC-based tool for function analysis using hierarchical function modeling based on the Air Force IDEFØ Function Modeling Method.
AI1™	An interactive PC-based tool for information analysis using hierarchical information modeling based on the Air Force IDEF1 Information Modeling Method.
AI1X™	An interactive PC-based tool for database design using hierarchical data modeling based on the Air Force IDEF1X Data Modeling Method.
AI3™	An interactive PC/MAC/UNIX-based tool for process and object state analysis using the Air Force IDEF3 Process Description Capture Method.
AI4™	An interactive UNIX-based tool for structured design of object oriented applications using the Air Force IDEF4 Object-oriented Design Method.
MDSE™	An integrated system that provides design decision support for engineering model development. This system also provides the means for capturing and managing engineering analysis plans and their rationale.
ACCESS	A knowledge based design advisor for the planning and design of heating and air conditioning systems. The primary applications for this system to date have been within the automotive industry.
COOLSYS	A knowledge based advisor for the planning and design of internal combustion engine radiator and water pump designs.

COS — A knowledge representation and reasoning system with embedded constraint management tools specifically developed to address the needs of flexible persistent management of long lived reusable knowledge assets.

HPCAD — A C++ based toolkit for development of High Productivity CAD systems. This toolkit supports the development of solid and surface geometry modelers as well as specialized translators for interfacing with existing commercial CAD systems.

KBSI also offers the best available IDEF training. Courses include IDEFØ, IDEF1, IDEF1X, IDEF3, IDEF4, and IDEF5. In addition, we provide IDEF modeling support, model review services, and IDEF courses customized for a particular organization's needs.

Knowledge Based Systems, Inc. provides systems engineering and systems integration technical support to engineering, manufacturing, and government organizations. The principals at KBSI have extensive experience in IDEFØ, IDEF1, and IDEF1X, and other classic information system analysis and design methods. We are also experienced in developing custom planning systems and artificial intelligence systems for engineering and manufacturing.

The application experience of our team includes the following:

Intelligent Design Support Systems

Integrated IDEF Modeling Support Environments

Intelligent Tutoring Systems

Seamless CASE Environments

Manufacturing Process Planning

Part Feature Extraction and Shape Based Reasoning

Space Station Design Support Environment

Integrated Management Decision Support Systems

Integrated Programming Environments for Embedded Real-Time Applications

Assemblers, Operating Systems, and Graphics Utilities for Specialized Hardware

The twenty-five professionals at Knowledge Based Systems, Inc. are experienced in using a wide range of hardware architectures, including Lisp machines, PCs, Macintoshes, and a variety of UNIX workstations. We bring a strong background in specialized, knowledge-based software solutions to manufacturing and engineering problems in industry, and government organizations. KBSI also has a proven track record in aggressively moving research results into software product development. Knowledge Based Systems, Inc. is

currently working on projects funded by NASA, DARPA, the Air Force, Chrysler Motors, and the National Science Foundation.