

56 P.

**IDEF3 Technical Report
Version 1.0**

**Richard J. Mayer
Christopher P. Menzel
Paula S.D. Mayer**

**Knowledge Based Systems Laboratory
Department of Industrial Engineering
Texas A&M University
College Station TX 77843**

**Reviewed by
Michael K. Painter, Capt, USAF
Armstrong Laboratory
Logistics Research Division
Wright-Patterson Air Force Base, Ohio 45433-6503**

January 1991

(NASA-CR-190279) [RESEARCH ACCOMPLISHED AT
THE KNOWLEDGE BASED SYSTEMS LAB: IDEF3,
VERSION 1.0] (Texas A&M Univ.) 56 p

N92-26587

Unclās
G3/61 0086873

**IDEF3 Technical Report
Version 1.0**

**Richard J. Mayer
Christopher P. Menzel
Paula S.D. Mayer**

**Knowledge Based Systems Laboratory
Department of Industrial Engineering
Texas A&M University
College Station TX 77843**

**Reviewed by
Michael K. Painter, Capt, USAF
Armstrong Laboratory
Logistics Research Division
Wright-Patterson Air Force Base, Ohio 45433-6503**

January 1991

Preface

This paper describes the research accomplished at the Knowledge Based Systems Laboratory of the Department of Industrial Engineering at Texas A&M University. Funding for the Laboratory's research in Integrated Information System Development Methods and Tools has been provided by the Air Force Armstrong Laboratory, Logistics Research Division, AFWAL/LRL, Wright-Patterson Air Force Base, Ohio 45433, under the technical direction of USAF Captain Michael K. Painter, under subcontract through the NASA RICIS Program at the University of Houston. The authors and the design team wish to acknowledge the technical insights and ideas provided by Captain Painter in the performance of this research as well as his assistance in the preparation of this report. Special thanks goes to the IDEF3 design team whose names are listed below:

Dr. Richard J. Mayer
Dr. Christopher P. Menzel
Dr. Paula S.D. Mayer
Dr. Thomas Cullinane
Dr. Douglas D. Edwards
Martha S. Wells
Shashank Joshi
Thomas M. Blinn

Summary

This document describes a language for the representation of process and object state centered system descriptions. IDEF3 is a scenario driven process flow modeling methodology created specifically for these types of descriptive activities. IDEF3 is based upon the concept of direct capture of facts about processes and events in a form that is natural to domain experts in a given environment. This includes the capture of facts about the objects that participate in a process, facts about those objects, as well as the precedence, and causality relations between processes and events. The goal of IDEF3 is to provide a structured method for expression of 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:

1. Recording the raw data resulting from fact finding interviews in systems analysis and knowledge engineering activities,
2. Determination of the impact of an organization's information resource on the major operation scenarios of an enterprise.
3. Documentation of the decision procedures affecting the states and life cycle of critical shared data (particularly manufacturing, engineering, maintenance, and product definition data),
5. System design, design tradeoff analysis, and simulation modeling

Two modes exist within IDEF3, process flow description and object state transition description. A process flow description is intended to capture knowledge of "how things work" in an organization, e.g., the description of what happens to a part as it flows through a sequence of manufacturing processes. 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, as they are called, form the basic units of an IDEF3 description. The resulting diagrams and text comprise what is termed a "description" as opposed to the focus of what is produced by the other IDEF methods whose product is a "model".

An IDEF3 Process Flow Description captures a description of a process and the network of relations that exists between processes within the context of the overall scenario in which they occur. The intent of this description is to show how things work in a particular organization when viewed as being part of a particular problem solving or recurring situation. The development of an IDEF3 Process Flow Description consists of expressing facts collected from domain experts in terms of five basic descriptive building blocks. Each of these building blocks will be described and illustrated in subsequent sections of this document.

1.0 Introduction and Background

The original IDEFs were developed for the purpose of enhancing communication among people who needed to decide how their existing systems were to be integrated. IDEF0 was designed to allow a graceful expansion of the description of a system's functions through the process of function decomposition and categorization of the relations between functions (i.e., in terms of the Input, Output, Control, and Mechanism classification). IDEF1 was designed to allow the description of the information that an organization deems important to manage to accomplish its objectives. The third IDEF (IDEF2) was originally intended as a user interface modeling method. However, since the ICAM Program needed a simulation modeling tool, the resulting IDEF2 was a method for representing the time varying behavior of resources in a manufacturing system providing a framework for specification of math model based simulations. As a result, the lack of a method which would support the structuring of descriptions of the user view of a system has been a major shortcoming of the IDEF system of languages. The basic problem from a methodology point of view is the need to distinguish between a description of what a system (existing or proposed) is supposed to do versus a representative simulation model that will predict what a system will do. The latter was the focus of IDEF2; the former is the focus of IDEF3. There are two additional description capture IDEF methods under development: IDEF5 will be a method for knowledge acquisition, and IDEF6 will be a method for capture of information system design rationale.

The second class of IDEF methods that have been developed are focused on the design portion of the system development process. That is, they encapsulate the best known method for design with a particular technology (or class of technology.) Currently there are two IDEF design methods, IDEF1X and IDEF4. IDEF1X was developed to assist in the design of semantic data models. IDEF4 was developed to address a need for a design method to assist in the production of quality designs for object oriented implementations. IDEF4 like IDEF1X is intended to service the needs of the systems designers and programmer analysts who are building and evolving large information systems. Unlike IDEF1X which encapsulates a design method for relational database design, IDEF4 encapsulates the principles for design of object oriented applications and databases. Figure 1 illustrates the planned areas of application for the IDEF methods relative to an updated Zachman framework for information system architectures [Zachman 86, IUG 90, and Mayer et al. 89].





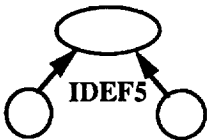
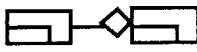
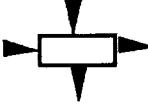
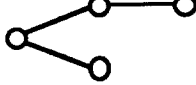
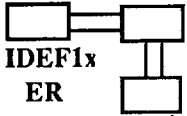
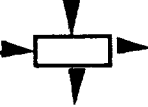
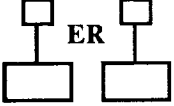
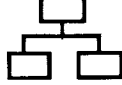




	DATA	USER	FUNCTION	NETWORK
OBJECTIVES/ SCOPE	List of Things Important to the business  BSP IDEF0 ENTITY = Class of Business Thing	List of Scenarios the User Performs  IDEF0 BSP	List of Processes the Business performs  IDEF0 CSF Process = Class of Business Activity	List of Locations in which the business operates 
DOMAIN MODEL	e.g. Concept Model  IDEF5 ENT = Bus. Con. ReIn = Association	e.g. User Role Description IDEF3	e.g. Business Process Descrip. IDEF3	?
MODEL OF THE BUSINESS	e.g., Entity / Relation Diagram IDEF1  ENT = Info. Entity ReIn = Bus. Rule	e.g. Organization Process Descrip. IDEF3	e.g., Function Flow Diagram  IDEF0	e.g., Logistics Network  Node= Bus. Unit Link= Bus. Relatn
MODEL OF THE INFORMATION SYSTEM	e.g., Data Model  IDEF1x ER ENT = Data Entity ReIn = Data ReIn	e.g., Transaction model IDEF3	e.g., Data Flow Diagram  DFD	e.g., Distributed System Arch ? Node=I/S Func. Link=Line Char.
TECHNOLOGY MODEL	e.g., Data Design  ER ENT = Segment ReIn = Pointer	e.g., Object Design IDEF4	e.g., Structure Chart  SCG	e.g., System Arch ? Node=Hardware Link=Line Spec.
DETAILED REPRESENTATIONS	e.g., Data Design Description  ENT = Field ReIn = Address	e.g., User Inter-Face Code 	e.g., Program 	e.g., Network Architecture 
FUNCTIONING SYSTEM	e.g., Data	e.g., Scenario	e.g., Function	e.g., Communication

Figure 1. Updated Zachman Framework

One of the primary mechanisms used for descriptions of the world is relating a story in terms of a ordered sequence of events or activities. The IDEF3 method for process description capture is designed to capture descriptions of the process flow and object state transitions associated with a particular situation (scenario). A scenario can be thought of as (1) a sequence of activities that constitute a particular process or event, (2) a set of situations that describes a problem in an organization or system, or (3) a process description in a given setting. The IDEF3 method is designed to be used by both the area expert and an analyst / modeler for capturing the knowledge of the area expert about how a particular process, event, or system works.

IDEF3 was also designed as a mechanism for recording facts that analysts capture either from direct observations or interviews within a particular domain. The method provides the capability for capturing the temporal (time-related) precedence and causality relationships about the process or event. IDEF3 uses the context of the scenario essentially to drive this description capture method by providing a boundary to what information the description will cover. The IDEF3 method presents the description capture within an environment that is natural for the area expert to use to describe his knowledge about his area. The purpose of such a description is to provide the data for:

- Determination of the impact of an organization's information resource on the major operation scenarios of an enterprise.
- Documentation of the decision procedures affecting the states and life cycle of critical shared data, in particular, manufacturing, engineering, and maintenance product definition data.
- System design and design tradeoff analysis.
- Determination of the pragmatic data models or how the semantics of data are used by the enterprise.

It should be noted that many of the uses of IDEF3 proposed in Figure 1 are direct extrapolations from uses of informal process flow modeling in the system development life-cycle as described in the early phases of the IDEF3 development activities. Much of the thinking behind IDEF3 has been heavily influenced by those experiences, particularly the experiments performed by Edwin I. (Sam) Nusinow of Knowledge Based Engineering Enterprises in his work modeling engineering data management and control processes. Similarly, many of the interesting ideas behind the semantics for junction

types that will be described later in this document were originally suggested by Timothy Ramey and Stu Coleman of Pacific Information Management, Inc.

1.1 Motivations behind the Development of IDEF3

The motivations which drove the requirements and development of the IDEF3 method address three basic needs.

- First, there is a need for a method to describe design data life cycle characteristics. That is, this method will describe how technical data are managed during the development stages and the life cycle of a product.
- Second, there is a need for a method to formalize external-constraint driven requirements definition methods. Such a method should (1) provide a description of the functional constraints on a system imposed by its operational environment and (2) communicate how the system will be used once in place.
- Third, there is a need for improved productivity in the system analysis model development process. The benefits of having this type of method would allow the system analysis model development process to be done faster and yet produce more correct, consistent models with the result of a reduction in the overall cost of enterprises analysis.

Each of these needs will be discussed in more detail.

The first major need was for a method to describe design data life cycle characteristics. One approach to a successful initial information integration strategy is to start with the *control* of the configuration of engineering design data within a manufacturing organization. To achieve this control, one must first map out (1) the design artifacts or the containers of design information, (2) the state transitions through which these artifacts proceed, and (3) the decision logic that is used to determine the state transitions. Since these transitions are largely affected by the activities of the associated organizations, this requires a definition of the activity sequence and decision making logic of the organization.

The second major motivation behind the development of the process flow method was the need for a method to formalize external-constraint driven

requirements definition methods. It has been repeatedly demonstrated in practice that an effective mechanism for the design of a new system is to describe an ordered sequence of the activities as they would be experienced in the organization within the context of a given scenario. This process description is then used as a framework for proposing alternative *external views* of possible information systems to support the activities of the organization within the given context. Such development (design) approaches have been referred to as External-Constraint driven design approaches. Once this framework is established, each organizational activity can be used as a context for describing *what* the application would do to support that activity from the *participant's* perspective. In the Air Force Integrated Design Support (IDS) program, the *what* was represented as a sequence of *man/machine* actions, where the machine presented a prompt (or a form to fill out) and the human responded with the appropriate action. The general method used in such an approach can be an effective requirements definition mechanism, particularly if combined with the IDEF0, IDEF1, and IDEF1X methods. IDEF3 was designed to be extended naturally to serve as a tool for such an external constraint driven design method.

The third major motivation behind the development of the process flow method was the desire to speed up the process of business systems modeling. The experience of the technical coalition who identified such a need was that 'description of process flow' is one of the most natural ways to introduce an organization expert to information system modeling. A proven interview method for systems analysts traditionally starts with the identification of the major problems or situations that the domain expert deals with daily. Using each of these as a scenario or focus point, the area expert can easily describe what decisions, activities, and events surround or are included in the normal processing of such a problem or situation. The design of IDEF3 capitalizes on this phenomenon by making the first step of the method the requirement to identify the scenarios. As stated in the Introduction, this has the effect of the scenario *driving* the resulting IDEF3 practice. The IDEF3 method provides a structured way to communicate this descriptive information. Once this descriptive information is captured, the resulting structure and information can be easily used as a framework for construction of the other IDEF3 models.

As a result of these motivations, the IDEF3 method must provide (1) the concepts, (2) syntax, and (3) procedures for building 'requirements descriptions'. These requirements descriptions of a system must be adequately detailed to determine if a system received is acceptable. This implies that the IDEF3 method must support the following descriptions:

- Descriptions of scenarios of organizational activities.
- Descriptions of the roles of user types in these organizational activities.
- Descriptions of user scenarios or user interaction with the information system at the user function level.
- Descriptions of what the system is to do in response to the user functions.
- Description and delineation of user classes.
- Declaration of timing, sequencing, and resource constraints.
- Description of user interface objects (menus, keywords, screens, displays).

The last requirement of interface description implies the need to incorporate either (1) a formal language for display description or (2) the ability of the method to incorporate by reference (or insertion) drawings or sketches of the desired interface. At this time, the IDEF3 design has taken the latter option.

1.2 Applying the IDEF3 Method in Practice (How does it fit in?)

IDEF3 will be used in a multi-layer environment including (1) the area expert / practitioner layer, (2) the analyst / modeler layer, (3) the systems design layer, and (4) the meta-layer or tool ***ric is the preceding 'or tool' correct***. Layers are differentiated by the type of information IDEF3 will provide the participants at that level. For instance, IDEF3 models will be used for generation of (1) reports by the area expert / practitioner layer, (2) simulation models by the analyst / modeler, and (3) computer language code in the systems layer. The IDEF3 method will be applied in a series of four steps: (1) Identify the scenarios. (2) Within a scenario, perform the modeling exercise by preparing a diagram of the process or event. (3) With the diagram produced, the method will require an elaboration in natural language text to support the diagram. (4) As necessary, decompose scenarios further and repeat the modeling activity for each scenario produced.

The IDEF3 method captures facts and constraints. Facts are stated in simple sentences that describe an operation in the description of an event or process within the scenario. One guideline governing the statement of facts is: *If not recording something is going to give a reader a problem understanding the description, then the modeler should record it.* Constraints are background conditions or preconditions necessary for the operation to occur. Sometimes the distinction between a constraint and a fact is difficult to determine, but

there are general guidelines for recognizing the differences between facts and constraints. In general, for example, if a statement contains a "negative" (words such as "not," "never," "no"), then that statement is a constraint.

IDEF3 will be used by the practitioner or area expert as a data collection tool. At this layer, the area expert will attempt to determine (1) what gets acted upon (put on the IDEF3 object list) and (2) the mechanism and causes for these actions. From there, the data collection will be used by an analyst / modeler to provide the graphical syntax from the descriptions.

1.3 Introduction to Situations and their Descriptions

The arguments presented in the previous sections were put forth for answering the question "Why a process flow description capture method?" Other equally important (and often asked) questions would include:

- Why did IDEF3 come out like it did?
- Why the focus on descriptions rather than idealizations?
- Why not adapt / extend IDEF0 to just add in activations?
- Why not adapt / extend IDEF1 to just add in event classes?
- Why not use a simulation modeling language (e.g., IDEF2) or a queueing network language (e.g., Petri nets)?

The purpose of this section is to address these questions by introduction of the theoretic influences behind the method. From a summary view, IDEF3 is very different in nature from existing system engineering modeling methods in that it was not designed as a modeling tool but rather as a language for the organization and expression of process descriptions. This description capture nature and many of the motivational concepts on which IDEF3 is based come from the pioneering work of Dr. Sijjr Nijssen and Mr. Max Wilson who first promoted the notion of an information system as the embodiment of a discourse situation between agents in an organization. The work of Dr. Jon Barwise who initiated situation theory and situation semantics was also a principal motivation of the concepts behind IDEF3. That theory 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.

1.4 Description versus Models

The history of process modeling is long and illustrious. However, process modeling requires skills and experience beyond those normally available to the individuals needing the analysis results obtained from process models.

It is important first to distinguish between models and descriptions. We emphasize that while models may well be constructed from descriptions, our task is not the construction of models, but the formal representation of descriptions and the information they convey.

To get at the distinction, a model can be characterized as an ideal system of objects, properties, and relations that is designed to imitate in certain relevant respects the character of a given real-world system. The power of a model comes from the ability of a model to simplify the real-world system it represents, and to predict certain facts about that system in virtue of corresponding facts within a model.² Thus, a model is, in a sense, a complete system. For it to be an acceptable model of a given or imagined real-world situation, it must satisfy certain "axioms" or conditions derived from the real-world system.

A model can be characterized as an idealized system that we design to have certain elements with defined properties within a particular structure. The general idea of model making is to look and see what other properties the model has beyond those we have designed. The power of the model comes from the ability of a model to predict properties (relations) which turn out to hold in the real world. That is, 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. The danger of modeling is in conflation of the real world with the model elements and structures we have designed to model that world. The difficulty in using models lies in the knowledge required of idealized systems that can serve as reliable predictors as well as the knowledge required to apply those reliable idealizations to a particular situation (to say nothing about the knowledge required to detect when the application has failed).

A description, on the other hand, is a recording of facts or beliefs about the world around us, recording our observations or beliefs. As such, descriptions are in general partial; a person giving a description may omit facts either because they do not seem relevant or by error. Thus, there are no preconditions on an acceptable description either for accuracy or completeness

² See (Corynen, 1975) for a detailed analysis of this phenomena.

and no 'axioms' to be satisfied. Descriptions are assumed to be true, but incomplete. In fact, one very powerful use of models is to fill in the gaps in our descriptions. The accumulation of descriptions is thus prior to and distinct from the construction of models. Generally, the conditions one puts on acceptable models are derived from descriptions one receives from domain experts; they are, so to say, the data from which models are built. The importance of descriptions is that they serve as sources of information to a model designer when observations are inconvenient or impossible. That is, it is unlikely that a systems analyst will actually witness all the phenomena that a person who has spent ten years or more in the domain will have witnessed.

Description formulation often employs abstractions. Abstractions are distinguished from idealizations (the stuff that models are made of) in that instances of abstractions are assumed to have realizations in the real world, whereas instances of idealizations are not restricted in this manner. One advantage of descriptions is that those working in the domain generally require less training to produce reliable instances of descriptions of their domain. A disadvantage of descriptions has been the lack of effective means of organizing those statements of facts. IDEF3 is proposed as one such organizing description capture mechanism.

Descriptions are thus essential to the model building process. An accurate treatment of such descriptions requires two components, one having to do with the descriptions themselves as linguistic entities, the *syntactic* component, and the other with their content and the information they convey, the *semantic* component. That is, on the one hand, it is important that there be an effective means of representing the descriptions themselves, a means of capturing their logical form. On the other hand, there must also be a rigorous account of their information content.

Languages to support the representation of models provide constructs for capturing the parameters of the idealized elements and the construction of systems of those elements. In comparison, a language to support the capture of process descriptions must allow for the formation of the abstraction elements needed for a particular situation and allow for the statement of facts about specific instances of those abstractions. In addition, description capture languages must also support the statement of facts about the relationships between abstraction elements and between possible instances of different abstractions. These requirements result in very different language structures between a modeling language and a description capture language. The following illustrates some of the differences between a process description and process models such as:

- Flow charting languages have (1) no capability to describe asynchronous behavior, (2) limited capability to support decomposition, (3) no capability to describe objects, properties, and relations between objects, and (4) no capability to describe enablement relations between processes.
- Network simulation languages have (1) limited capability to describe relations between objects, (2) some object and property capabilities, (3) no support for partial information description, (4) fixed elaboration of process descriptions, and (5) no decomposition description relations.
- Simulation programming languages (i.e., simulation languages that either are full-fledged programming languages or allows entry into a programming language) generally have no primitive support for (1) decompositions, (2) enablement relations, or (3) partiality. However, while they could be used to implement a description capture system for the language we have defined, there would be no advantage in doing so and many disadvantages.

Any well-formed language consists of both a syntax including a vocabulary and grammar and a semantics including a method of ascribing meaning to statements formed in that language. The above discussion of differences between a modeling language and a description capture language has focused primarily on syntactic differences. However, these syntactic differences are indicative and sometimes inseparable from more basic semantic differences. To achieve the predictive power of an idealization, the interpretation of statements in a modeling language is restricted to rigidly defined mathematical structures which do not consider context of use. That is to say the 'real world' interpretation of a system described in such a language is left to the modeler.³ Unfortunately, this limits the amount of information that can be communicated with such a language. A description capture language must provide for the capture of this 'context of use'. It also implies that such a language must have a semantics that is designed to or at least attempt to take into account the mapping between statements used to form a description and the human interned understanding of the situation so described in the statements of that language.

³ This is not a criticism of such languages, since it is the source of proven power. In fact, the discovery of such context independent mathematical structures is a driving force behind most research. This context independent interpretation contributes to the "efficiency" quality of a modeling language; which is to say that the same model can be used in a predictive capacity in many situations (think of the number of phenomena that are modeled with simple first order differential equations).

1.5 Need to Facilitate the Flow of Information

A basic objective of any system engineering method is the facilitation of the flow of information between two or more situated agents including:

1. The domain experts and their management,
2. The domain experts and the systems analyst,
3. Systems analysts with different areas of responsibility,
4. The analysts and designers,
5. Designers and lead programmers,
6. The lead programmers and their programming staff.

Each of the above mentioned roles as well as many others has a part to play in the evolution of an existing system or the development of a new system. Depending upon the role and the particular circumstances of the communication discourse, the information transferred has different effects. An analyst experienced in certain problem situations and technology capabilities needs accurate descriptions of the 'AS IS' environment. Even if his services are merely engaged to review the results of an ad hoc design, he must obtain knowledge about the current situation in order to interpret his experience. The analyst must communicate the constraints that must be met by the new system for it to achieve its goals. The designer must obtain information about both the existing system that he must interface with and the constraints that the system must operate within. Thus, the same sentence or model carries different meanings for different people at different times in this development process.

1.5.1 Communication and the Flow of Information

In each of these communication encounters, the agents involved are communicating information about a piece of the world that they have experienced or find themselves in. The name we use to refer to that carved-out piece of the world is a **situation** [Devlin 91]. Communication between agents is viewed as process of information flow between situated agents. The acquisition of such knowledge, belief, or observation is tied to the perception of the parts of the world that the agent is attuned to. As in [Devlin 91] "Information flow is made possible by certain (what will be called) constraints

that link various types of situation." Attunement to such constraints then is a prerequisite for information flow and hence for communication between situated agents. The design of IDEF3 can be viewed as an attempt to provide an organizational framework for the capture of descriptions of situations and the constraints which will allow for the efficient communication of those situations to the agents involved in the system development process.

The early IDEFs met this challenge of enhancing communication by intuition and experience of the development team without a formal underpinning. IDEF3, however, has been designed with both a basis of a strong underlying theory as well as the intuition and pragmatic experience of the development team. We have taken advantage of emerging theories of knowledge and the flow of information, specifically Situation Theory [Barwise and Perry 83]. IDEF3 was designed from theory to practice rather than the other way around with the anticipating of a resulting method which will have better first-time performance.

The previous discussion of the motivation behind the definition of a new method for process flow and object state description points to two specific needs. One need is for a method to support the development of a mechanistic description of *how* an organization solves a particular type of problem. Another is the need to describe *how* an existing or proposed information system supports or will support the organizational process. Additionally, a third need is for a method to support the description of what the 'interaction' will be between agents and the information system. In the past, we were primarily concerned with human agents or users. In an integrated environment, we must recognize that a significant class of agents will be other information systems.

1.6 Understanding the Common Sense Notion of "Process"

To understand the idea of capturing process descriptions, it is first necessary to know what we mean by a process. We are not using the term in a technical sense, but rather in an ordinary language sense, keeping with IDEF3's role as a methodology for acquiring the intuitive knowledge of domain experts. Unfortunately, the term 'process' is quite ambiguous in English. Therefore, we will refine our understanding of process terminology in ordinary language before we can characterize the intended sense of 'process' more clearly.

Since we are interested in capturing a human's understanding of the world around him and how it works, it is necessary to characterize the concept of a

process in view of that understanding. Such a characterization is bound to be difficult since the notion of 'change' is basic to the concept of process. Intuitively, the term 'process' is used to describe an isolatable event or occurrence. As such, it can be assigned a more or less definite starting point, typically associated with the satisfaction of certain antecedent conditions, and continue indefinitely. A process will, in general, involve objects with certain (perhaps changing) properties standing in specified (perhaps changing) relations. A process can also stand in relation with other processes, e.g., a process can start, suspend, and terminate other processes. That is, objects or information about objects can be shared between processes. In addition, one process can change the properties of such a shared object and cause the exclusion of another process execution. One interesting characteristic of the concept of process is the lack of a physical perception of a process. One can observe the objects and their properties or relations, but not the process itself. Possibly as a side effect of this phenomena, the concept of a process has a well established type versus instance distinction which is so accommodated by the language as to be obscured. For instance, we previously noted that a process can start another process. Strictly speaking, this means that that an instance of one process can start an instance of another. This distinction between type and instance turns out to be central to much of what is difficult in the capture and representation of process descriptions. It also strongly affects the reasoning with or general processing of such descriptions.

There are a number of problems associated with any attempt to describe change. Some of these problems are definitional in nature, such as defining terms as 'process', 'event', and 'activity' as well as characterizing the difference between these concepts. Presuming a consistent definition, the next stumbling block comes from the difference between the orderings and relations which can be established between *types* of such concepts and those which apply to *instances* of such concepts. For example, simple relations like *before* which can unambiguously be applied to event instances (e.g., I ate supper before I typed this section) require considerable elaboration if applied at the *type* level. If one activity type precedes (follows, overlaps, etc.) another activity type, it may not necessarily imply that all possible instances of one activity type precede all possible instances of the other. More often, it means that the instances of the two activities are pairwise related such that each instance of one activity precedes a corresponding instance of another activity. In fact, it is often the case that an even stronger relation is implied, where an instance of the first activity instance in the pair causes an instance of the second to occur. Such ambiguity in type-based activity descriptions can be resolved with additional specification mechanisms although there is a trade-off of understandability versus accuracy. Unfortunately as the level of specificity increases, there is a corresponding decrease in the comprehensibility of the

models. Traditionally, the basic situation has been an untenable one; descriptions of large systems could either be accurate or comprehensible. With IDEF3, however, a structure is provided that can tolerate partiality in the description allowing both comprehensibility and accuracy.

It is crucially important to distinguish between 'process types' and 'process instances', or 'individual processes'. Indeed, it is important to distinguish generally between types and instances with regard to many other kinds of entities as well. We think of an **individual** process as a definite occurrence with a specific time and place. Process **types** may be thought of as **classes** of individual processes or **properties** that individual processes may have. It is unfortunate that the English language does not distinguish between process types and individual processes: The word 'process' can refer to either. For example, consider a constraint of "A request for change must be logged in before it can be presented at a meeting of the configuration control board." This constraint does not refer to the class of objects *request(s) for change* or the *logging-in* event type or the *meeting* event type. These references do not point to a particular object or event, but rather they refer to a **prototypical** object or event. A prototypical object or event refers to the general type of events and not a particular object that exists or an event that has occurred. IDEF3 uses the term **indeterminate** to identify a prototypical object or event. An actual particular instance can be differentiated from an indeterminate by providing characteristics of the particular instance that provide a definite temporal or spatial reference. The term 'type' is appended to an event or activity reference to refer to this general group of prototypical events or activities.

Unfortunately, many of the relations and constraints to be described in a process flow description refer to or involve instances of these indeterminate types of a request for change and an instance of the *logging-in* event type and an instance of the *meeting* event type. Therefore, IDEF3 method provides the means for talking about instances, indeterminates, and types. In the following sections of this report, we will often use the term 'instantiation' to refer to an instance of a particular indeterminate or prototypical event or state of affairs. We will also use the term 'anchor' to refer to a set of instantiations which adhere to the stated constraints associated with a particular process description. This distinction between 'instances' and 'type' actually appears in many systems engineering methods. For example, in IDEF1, we model entity-classes but all the rules (such as the No Null and No Repeat Rules) operate on individual members (or instances) of entity classes.

In IDEF3, however, we will attempt to maintain the distinction rigorously whenever it matters. It does not always matter. For instance, it does not

matter in very general contexts (as above) or where the term 'process' occurs as an integral part of phrases like 'process flow description capture' or 'process model'. Process types may vary from general to specific. An individual process p which is among those picked out by a process type P will be called an **instance** of p . If process types P and Q are such that any type instance of P is also used in an instance of Q , then P is said to be a **subtype** of Q . Similar terminology is used for types and individuals of other varieties than processes.

On important note about the more general use of the term **individual**: not all individuals are concrete entities. Some types, like **number**, may have instances which are abstract entities; these instances are nonetheless individuals. In fact, the term 'individual' is really more or less synonymous with 'instance'; a thing called an individual only with a tacit reference to some type of which it is an instance.

1.6.1 Narrow and Broad Senses of "Process"

Another problem in describing processes in English arises from the language's intense focus on the details of temporal succession, characteristic of IndoEuropean languages.

There is a sense of 'process' in which the word is distinguished from 'event', 'state of affairs', 'eventuality', 'occurrence', or any number of other words of this general class. In fact, there are several such senses of 'process', each stressing a different kind of distinction between processes and other things of this general kind. For instance, in one sense processes are supposed to have internal structure. An example of this sense would be where each instance of a given process type is supposed to be divisible into temporal subparts which are process instances of the same process type, whereas this would not be true for certain other conceptions of process. The linguist and philosopher Zeno Vendler [Vendler 67], among others, has gone into great detail in classifying things of this kind, coining individual terms for concepts represented by variations in meaning in English.

On the other hand, there is another sense of 'process' synonymous with 'event'. In this sense, the extensions of any of the other terms listed in the preceding paragraph would be subsets of the class of processes; a process is **anything** of the general kind described in the preceding paragraph. It is this broadset sense of 'process' with which we are concerned in IDEF3; the Vendlerian classification is irrelevant to our purposes. IDEF3 has its own ways of distinguishing one kind of process from another based on the internal structure of the instances. The precise technical term we have invented for

processes in this broad sense is 'unit of behavior' or UOB, which simply means a process or event, in the most general senses of those terms. We continue to write simply 'process' where we feel it will be clear that any UOB is meant, not just a process in some narrower sense.

Besides the type instance issues there are other issues including:

- Representation of a situation view (some arbitrary complex set of objects with properties standing in relations within some space time bounds). We use the term 'situation view' because a different observer might and almost always will describe a different set of objects, properties, or relations at the same space time boundary.
- Changing situations in which nothing is happening except time moving forward (states of affairs, so to speak).
- Changing situations in which what is happening may be happening too quickly or slowly to perceive (i.e., glue drying).
- Models of time, time granularity, and measurement systems (the tau problem) and issues of ordering things in general.
 - Concurrency versus sequentiality.
 - Causality versus enablement versus coincidence (an issue of descriptions or judgments made on descriptions).
- Common sense notion of object persistence (if it is not mentioned then it does not change, e.g., occurrence of the same free variable across boxes). Conservation of objects contribution to the frame problem.
- Call no waiting, call waiting, the assumed return to a previous state. That is, issues that particularly arise in describing man/machine interaction processes.
- Complexity of possible relations. (The result being to build your own links.)
- Instancing issues, i.e., rates.

1.6.2 Situation Based Descriptions of Processes

The manner in which this tolerance is achieved can best be understood by further exposition of the Situation Theory which motivated the IDEF3 design. As mentioned previously, **situations** name pieces of the world that an agent is both *a part of* and *attuned to*. A real situation is made up of real objects

standing in relations at some space-time location. It is important to note that situation theory does not comment on the way the world really is. That is, it does not suppose that the world is actually broken up into situational pieces, only that our abilities to perceive the world around us are limited, and that this 'reach' is reflected in our languages and reasoning methods. This is a radical departure from traditional theories of logic and natural language semantics. Those wishing to pursue the extent and implications of this departure should look to [Barwise 83 and Devlin 91]. Because **situations** are countenanced as perceptually based partitions of the world surrounding an agent, their descriptions must capture the perceptions of the agent at a location in space and time. That is, they must have symbol set and structures for representing objects, relations, and spatio-temporal intervals.

It is best to think of these descriptions as abstract situations (**a-situations**) corresponding to their real world situations (**r-situations**). The a-situations have several interesting properties. One of those has to do with whether or not the a-situation is a factual description of the associated r-situation. That is, if the description contains a reference to some objects standing in some relation in the a-situation, do those objects stand in that relation in the r-situation? If so then the a-situation is said to be factual. There are two other key notions behind the a-situations. One is that they need not be 'complete' (sometimes labeled 'actual' in the technical literature), meaning that there certainly will be objects and relations at the spatio-temporal location which never get mentioned in the a-situation. The other characteristic is that a-situations are themselves first class-objects in the situation theory, implying that they can stand in relations just as the elements of the r-situation that they describe. This is a key property of the theory in that it allows for the formulation of a notion of constraints as distinguished relations between types of a-situation descriptions. It so happens that attunement to such intersituation constraints can be used as a powerful mechanism for explaining the flow of information between two situated agents.

The internals of an a-situation are represented by a language. In the IDEF3, this language includes both a graphical and a linear text form. The key elements of this language are symbols standing for:

- objects in the r-situation or other situations
- relations between objects
- spatio-temporal locators
- polarity indicators
- indeterminates (variables)

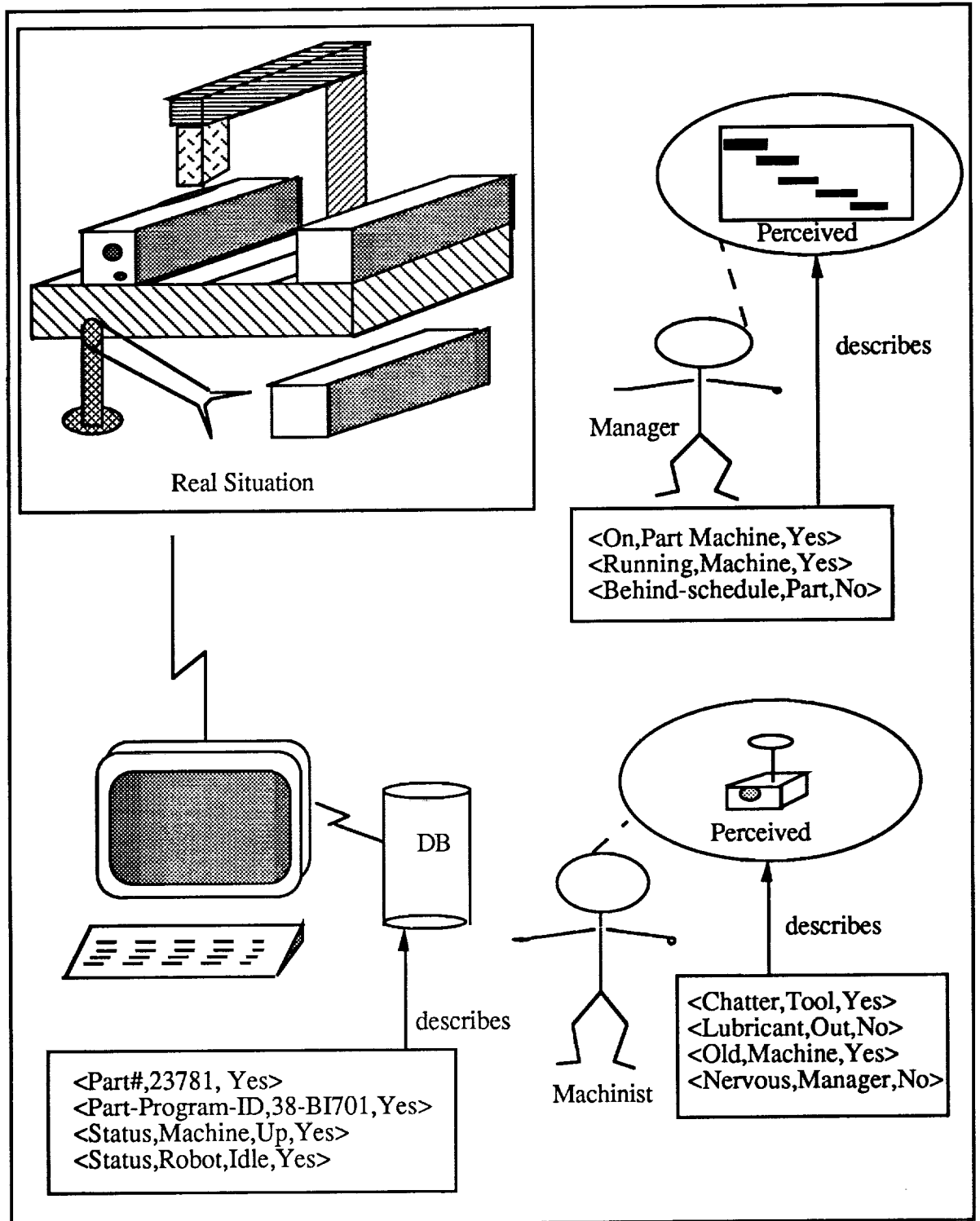


Figure 1.2. Real versus Abstract Situations

With these primitives, one can build up higher level constructs of an a-situation description language. For example, the expression including a

relation symbol and a set of objects (denoted $\langle r, o_1, o_2, o_3; i \rangle$ where r stands for a relation and the o_j 's stand for objects) is called a **constituent sequence**. An example of such a constituent sequence might be $\langle \text{on, part, conveyor} \rangle$. Similarly, a constituent sequence including an indicator of whether a particular set of objects stands in a particular relation (denoted $\langle r, o_1, o_2, o_3; i \rangle$ where r stands for a relation and the o_j 's stand for objects and the i stands for a polarity) is called a **state of affairs** (also sometimes referred to as an **infon**) An example of such a state of affairs expression would be $\langle \text{changed, part, dimension; yes} \rangle$. Collections of states of affairs with an associated spatio-temporal locator make up the most elementary a-situation descriptions. For example:

(here, now, $\langle \text{typing, ric, document; yes} \rangle$
 $\langle \text{sitting, ric; yes} \rangle$
 $\langle \text{operational, Macintosh; yes} \rangle$)

Since an a-situation is itself an object in a situation theory language, collections of such objects form complex descriptions referred to as **courses-of-events**. There are circumstances under which it is convenient to form collections of courses-of-events. Such collections are referred to as **event-schemas**.

Referring to the type token or type instance problem previously discussed is a good way to understand the next level of a-situation descriptions, those which involve indeterminates. Since a-situations are a part of the medium for information flow, situation theory provides for some of the elements in a description to have an indeterminate reference. For example, it is one thing to describe the situation of 'my car broken down on a particular day at a particular spot on a remote Texas country road', but we often want to describe the more general situation (of being 'broken down on a lonely country road') where the specific person place and time are unspecified (indeterminate). Situation theory provides for this through the use of indeterminates standing for objects, relations, or names. We can use an indeterminate in a situation description to form what is called a **role** which corresponds very nicely to the common sense notion of role. For example, the role of the machinist in a machining course of events or the role of the initiator in a transaction course of events.

As mentioned earlier, the last important type of expression is the 'constraint'. Constraints are expressions that include conditions in elements of an a-situation description that contains indeterminates or that include 'involvement relations' between courses-of-events. Constraint expressions serve to tie

situations together into structures that describe (if you will, larger situations) the agent's perception of how the individual situations fit together. For example, the arrival of a damaged aircraft involves the situation of initiating a flight discrepancy report (or more intuitively, that the situation of kissing involves touching).

In the IDEF3 method, the graphical syntax is focused on the description of courses-of-events. The description of the actual objects, relations, and constraints are relegated to elaborations associated with the course-of-events that are expressed in the Information Systems Constraint Language (ISyCL) [Decker and Mayer 89].

2.0 Syntax and Semantics

2.1 Basic Elements of IDEF3 Descriptions

The basic building blocks of IDEF3 are Units of Behavior (UOBs), Junctions, Links, Reference Pointers, and Elaborations. Each of these components of a description can be represented by variations in the formats of box-arrow layouts and list type documents. The following sections describe the basic elements of the process description language of IDEF3; combinations of these elements are used to form precise descriptions of organizational systems. An IDEF3 process description captures a network of relations between actions in a specified scenario.

The basic symbol set of the IDEF3 process description language is displayed in Figure 2.1. At the top of this figure, the Unit of Behavior (UOB) box is used to represent complex states of affairs (normally courses of events or event types as described above). These UOBs are organized into a network structure where the connections represent constraints between the event types referenced by the box label (see Figure 2.2). Though not displayed in Figure 2.1, each UOB element in a process flow model carries the following attributes:

- Name (unique across the process flow).
- Label (displayed with the symbol).
- Text (a glossary entry).

The UOB node also carries with it the specification of a node number and an optional IDEF0 cross reference. The rest of the graphical syntax elements in Figure 2.1 including the junctions, links, and off-page references can be viewed as shorthand notations to ease the specification of the constraints between the UOBs. The solid links are shorthand for a precedence constraint between two UOBs. The dashed links are used to indicate to the reader that a special constraint between two or more UOBs have been specified and should be examined by the reader. The dashed link has no default semantics and must be queried for its meaning. The junctions are shorthand expressions for logical and timing dependent combinations of precedence constraints. The reference pointers are used to indicate off-page connectors or references to elaborations (object relation descriptions or constraint specifications). Each of these model elements will be described in further detail in the following sections.

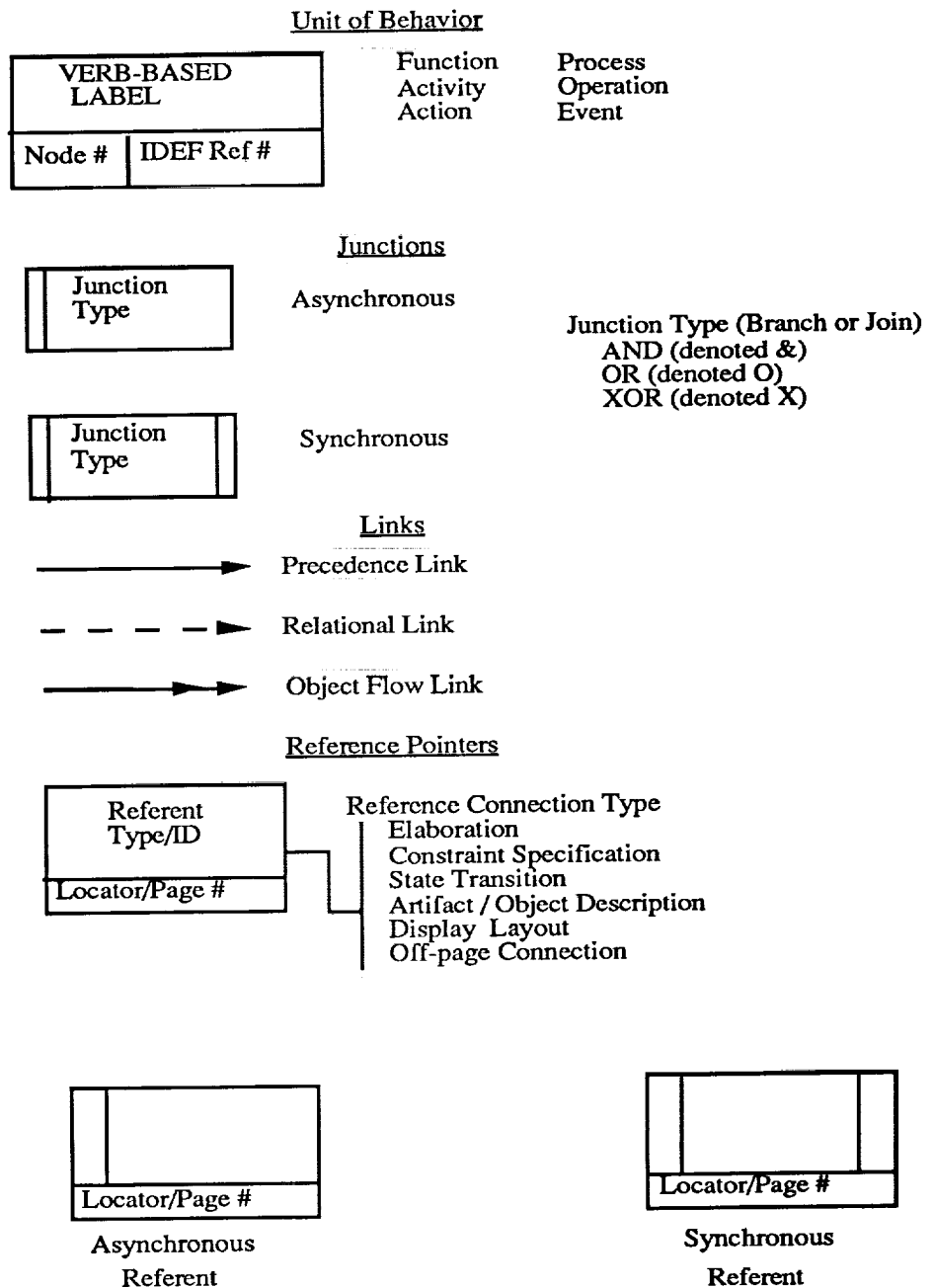


Figure 2.1. IDEF3 Process Flow Description Lexicon

2.2 Scenarios

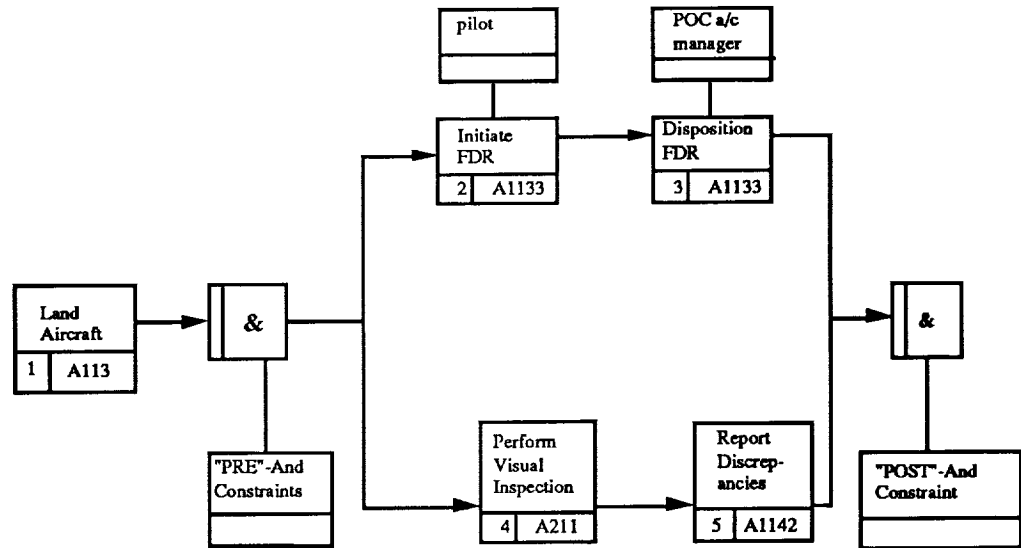
The basic organizing structure for IDEF3 process flow descriptions is the 'scenario' which provides the focus and boundary for the process description. A scenario is simply a recurring problem or situation within the organization being described that is used as a reference for organization of the process flow description. A 'scenario name' is simply a text string which names or identifies a typical problem solving activity or recurring situation within an

organization. Scenarios may often be taken directly from activities in an IDEF0 model (if one exists). The following are examples of typical process flow scenarios:

- Develop Die Design for Automobile Side Aperture Panel.
- Process Customer Complaint.
- Process Engineering Change Request.

A scenario name must be an action verb or phrase. It should be specific enough to allow the readers / description reviewers / authors to make judgements about the appropriateness of the content of the associated process diagrams. As well as a name, a scenario has associated with it a glossary which serves a role similar to that of the Context Statements in IDEF0. Unlike the Context Statement in IDEF0, an IDEF3 description will normally contain a number of scenario descriptions.

An IDEF3 diagram associated with a Scenario consists of a set of UOBs and constraints (special relations) among the UOBs. Figure 2.2 illustrates a typical scenario diagram in the IDEF3 syntax. The semantics of the links will be described in a later section. Since IDEF3 is representing a *description* of an organization or system it must be *partial*. That is to say, no claim is made by the modeler about the completeness of the description. What claim is made is that the description that is provided (in the set of UOBs, their associated links) is *factual* (i.e., the objects and relations described in the individual UOBs do in fact exist in the real world and stand in the prescribed relations.) This is in contrast to other types of models (e.g., IDEF2, Petri nets, and IDEF0) where the models represent an *idealization* of the real world and are assumed complete.



SCENARIO: PROCESSING DAMAGED AIRCRAFT

WITH
IDEF0 MODEL CROSS REFERENCES

Figure 2.2. Example IDEF3 Description

A partial reading for the scenario diagram presented in Figure 2.2 would be: Processing of a damaged aircraft begins with the successful landing of that aircraft. The completion of this (somewhat critical) activity initiates two independent activities, one being the initiation of the flight discrepancy report (FDR), the other the performance of a visual inspection of the aircraft. Note that since this is both asynchronous and a junction, an instance of both following events will happen but that nothing can be stated about the coordination of these events. Note that with the referent boxes, one can specify additional constraints even on the standard junction box semantics. Note also that distinguished roles (such as the agent roles of the pilot and the POC a/c manager) can also be emphasized by the use of such referent boxes. Otherwise, the objects participating in these UOBs are identified in the elaboration language statements mapped to each UOB.

2.3 Units of Behavior

In the capture of a description of "What's going on" within an organization or any complex system, there are a number of natural language concepts that must be accounted for, including:

- function
- activity
- action
- act
- process
- operation
- event
- scenario
- decision
- procedure

Each of these concepts is used/referred to in common language to describe 'states of affairs' and change in the world around us. In the design of IDEF3, a choice needed to be made about which of these objects referred to by the terms above would be explicitly represented. During the design of IDEF3 it was noted that each involves some 'circumscribed' behavior. That is, when someone refers to the *Planning activity* or *Make / Buy decision* or the *Contract award event*, that person is carving up the world around us into chunks of time (and generally space as well) to allow the description of what's going on in that chunk separate from the rest of the world. Therefore in IDEF3, a decision was made to provide a generic 'Unit of Behavior' concept which can be used to represent any of the above listed states of affairs or states of change. Whether a *Unit of Behavior* is classified as an 'event' or a 'process' or a 'function', etc., is left to the analysis of the description of that UOB and the structure surrounding it.

Each UOB in an IDEF3 process flow description is denoted by a box. The label inside the box is the 'display-name' of the UOB. Associated with the UOB is a unique 'name' which is formed out of a verb or verb phrase just as in IDEF0. Each UOB can have associated with it both 'descriptions in terms of other UOBs' and a 'description in terms of a set of participating objects and their relations'. We refer to the former as 'Decompositions' of a UOB and the latter as an 'Elaboration' of a UOB (see Figure 2.3). IDEF3 further distinguishes between decompositions which account for all the objects of the parent UOB and those which either leave out or add to the participating objects. The first variety are referred to as the 'Objective View', the latter as just 'Views' (see Figure 2.4). Multiple views are allowed in IDEF3 primarily because it is meant to be used in a description capture mode. Experience with IDEF0 has demonstrated the need to capture different views of the same activity. Unlike IDEF0, IDEF3 preserves the various views for latter use.

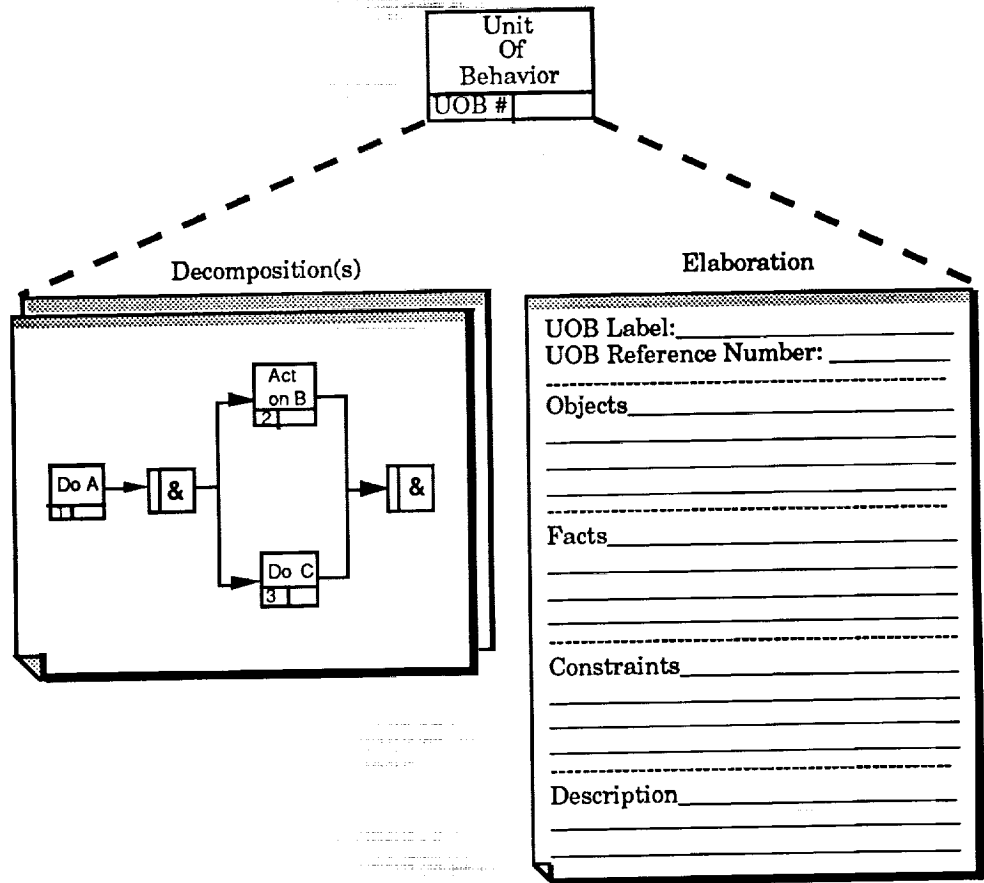


Figure 2.3. Descriptions of a UOB

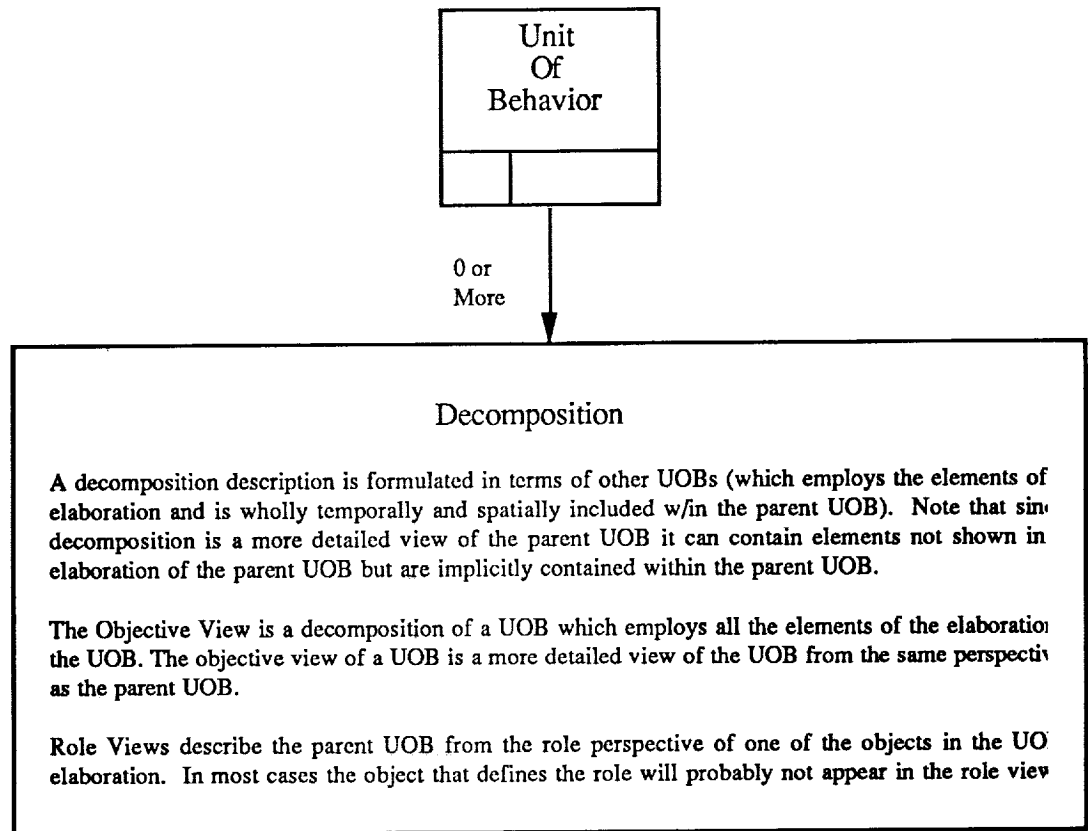


Figure 2.4. Views as Decompositions

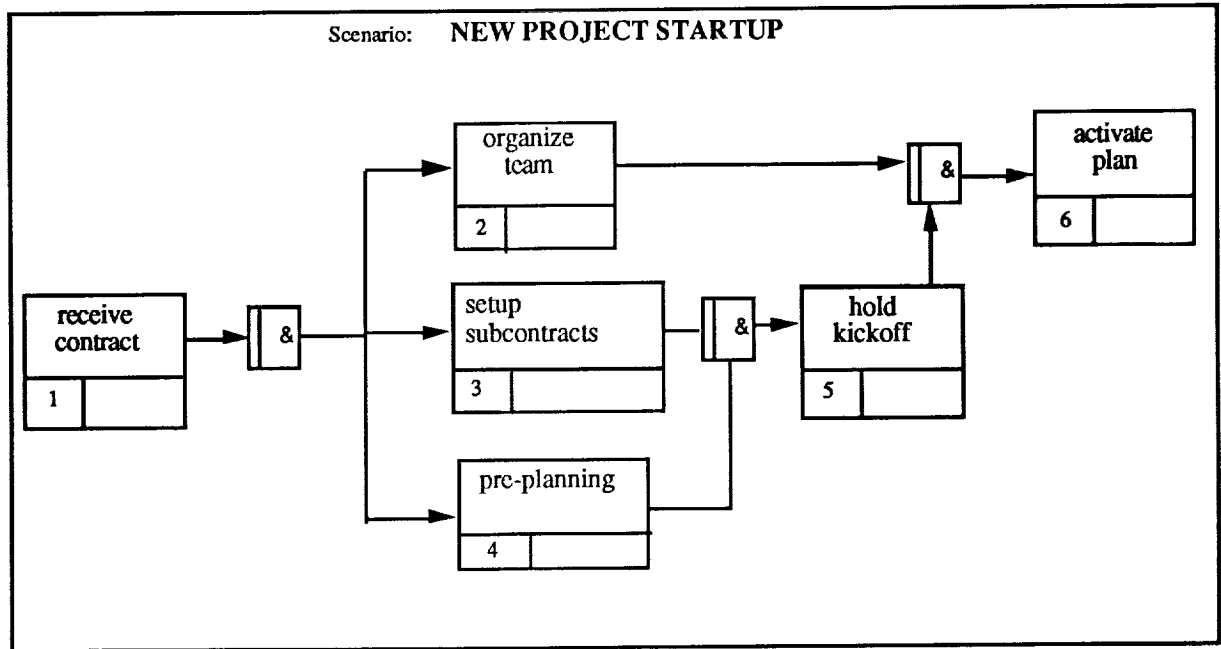
Since the links contain only constraints, one important point to note is that the UOBs are the focus for the description of the time consuming elements of a process description. Again, because of the description capture focus of IDEF3, it is entirely reasonable for UOBs to exist without any other links to any other UOBs. The interpretation of such free-standing UOBs is that such a referred to state of affairs exists, but nothing more is known about its connections with the remainder of the UOBs in the diagram in which it appears. Such a phenomenon may appear ill formed to modelers who are used to completely connected idealized models which form approximations of what the domain expert knows. Experience with knowledge acquisition of domain experts (or even the traditional interview process for information systems analysis) has shown the predominance of the partial view of an agent in the domain of interest. Simply, it is often the case that an agent observes or knows about far more than he (or she) can explain. IDEF3 allows the direct representation of the partiality of such descriptions (though in the formalization of IDEF3 a set of criteria for 'minimal computational' completeness is provided.

Each UOB in an IDEF3 process flow description is assigned a number to assist in the identification of that UOB for reference purposes and to establish

traceability with the IDEF0, IDEF1, IDEF1X, or other systems engineering models. During the development of the process flow description the UOBs are numbered sequentially by order of creation using the same scheme as is employed in IDEF1.

2.4 Elaboration Specification Language

IDEF3 allows for three levels of elaboration language specification. The first level is designed for use by the area expert, the second level by an analyst/modeler, the third level by software system developers. There is a fourth level called the meta-level which is used to establish or tailor the languages in the other three levels. A second level UOB elaboration is expressed in the IDEF3 situation description language. An example of such a specification is given in Figure 2.5. This language is a standard first-order language of the sort laid out in [Menzel 89a]. At the time of this report, an IDEF3 domain layer of the IISyCL is being defined that can serve as the formal recommended elaboration language. One of the features of this language is that it supports the naming or description of specific objects, places, times, and relations as well as the naming or description of indeterminate objects, places, times, and relations (intuitively, variables). This allows the IDEF3 modeler to describe easily both specific UOBs and also UOB types. A UOB type is defined as any UOB whose elaboration contains an indeterminate. It should be noted that in practice most of the UOBs contained within a model will be UOB types rather than specific UOBs. This is a consequence of the way people describe how something works or what goes on in an organization or a system. In fact, a specific UOB description is so rare that giving it a special symbol (like a box with rounded corners) was considered.



Elaboration: **NPS/ U5: Hold Kickoff**

(<at lab, during t,
 <attending, subcontractors, {x | subcontractor(x)}; yes>
 <attending, project_monitor, a; yes>
 <leading, project_manager, b; yes>
 <participating, project_team_members; yes>
 <draft project_plan; yes>
 <final project_plan; no>
 <allocated, task_assignments; no>>)

-- constraint on Overlapping Kickoff Meetings --
 before init of Hold_Kickoff kick_off1 ()
 "The project manager of a project can't be in two kickoff meetings simultaneously"
 [for_all x of UOB:Hold_Kickoff
 ((project_manager(x) = project_manager(kick_off1)
 and
 not (during (T(x), T(kick_off1)) or
 overlap (T(x), T(kick_off1)))))]

Figure 2.5. Example of a Specification

In the elaboration, a special classification structure is provided for the participating objects. Each object can be tagged as an 'Agent' if that object is considered to be the effector of the UOB. Or an object can be tagged as 'Affected' if the relations to that object are created or changed by / during the UOB. Or an object can be tagged as a 'Participant' if no causality or transformation is associated with that object as a part of the UOB description. Finally, an object may be tagged as 'Created' or 'Destroyed' by a particular

UOB. These classifications are optional. However, if supplied, they allow for automated analysis against the formal semantics [Menzel 89b].

The first level IDEF3 elaboration language is intended to be captured on an elaboration form (see Figure 2.6). An elaboration form captures the information from the area expert in natural language textual descriptions and presents this information in a structured manner. This elaboration form includes (1) an object list, (2) a fact list, and (3) a constraint list. The format for the elaboration form will include a pseudo natural language form of facts and constraint frames that the area expert can fill in. These sentence frames will be based upon example verb phrases and caseframes associated with these verbs. Any natural language text in the elaboration form is composed of simple sentences to reduce the sentence complexity and possibility of ambiguity.

Objects on the object list are those things or individuals that participate or are present in the occurrence of a unit of behavior. Objects are common across units of behavior, but perceived in different ways during the process being modeled.

Facts reference objects only if these objects are on the object list. Facts are supposed to be descriptive and make a single point per situation. Facts will often describe features or attributes of objects and represent information that is known to be true.

The constraints in an elaboration describe (1) the inter- and intra- process occurrence conditions, (2) the conditions under which an occurrence of the process can happen, or (3) the conditions under which an occurrence can be stops.

IDEF 3 Elaboration Form	
UOB Name: _____	UOB Reference Number: _____
UOB Label: _____	
Object Set:	
_____	_____
_____	_____
_____	_____
_____	_____
Fact Set:	

Constraint Set:	

Figure 2.6. Example of a Specification

2.5 UOB Decompositions

In IDEF3, a UOB can have many decompositions. This allows for the capture of different ‘perspectives’ of ‘what’s going on’ in the UOB itself. These perspectives are referred to as ‘Views’. There are two major types of Views, the ‘Objective’ view (of which there is only one) and the ‘Role’ view (of which there can be many). In the Objective view, all of the objects and relations that participate in the focus UOB are accounted for in the decomposition of that UOB. The Objective view may be thought of as a composite of all of the Role views. In a particular Role view, some of the objects or processes in the focus UOB elaboration may not be visible. Figures 2.7 and 2.8 illustrates the Objective and Role views of a *Hold Kickoff Meeting* UOB associated with an

Initiate Project scenario, illustrated in Figure 2.5. The specific meaning of the link types and junctions will be described in the next sections of this report. However, the basic reading of the first diagram is that a new project start-up involves the receipt of a contract followed by a number of activities in parallel. After the subcontracts have been established and an initial project plan is developed, a kickoff meeting is arranged and held. The results of this meeting combined with the team organization actually enable the initiation of the work on the project. The elaboration provided in Figure 2.5 is associated with the *Hold Kickoff Meeting* UOB. Notice in this elaboration the provision for constraint specification. In this case, we have provided both an English language version of the constraint as well as an ISyCL specification. It is expected that the area or domain expert would document the natural language version first and then, with the assistance of a modeling specialist, formulate the formalized version.

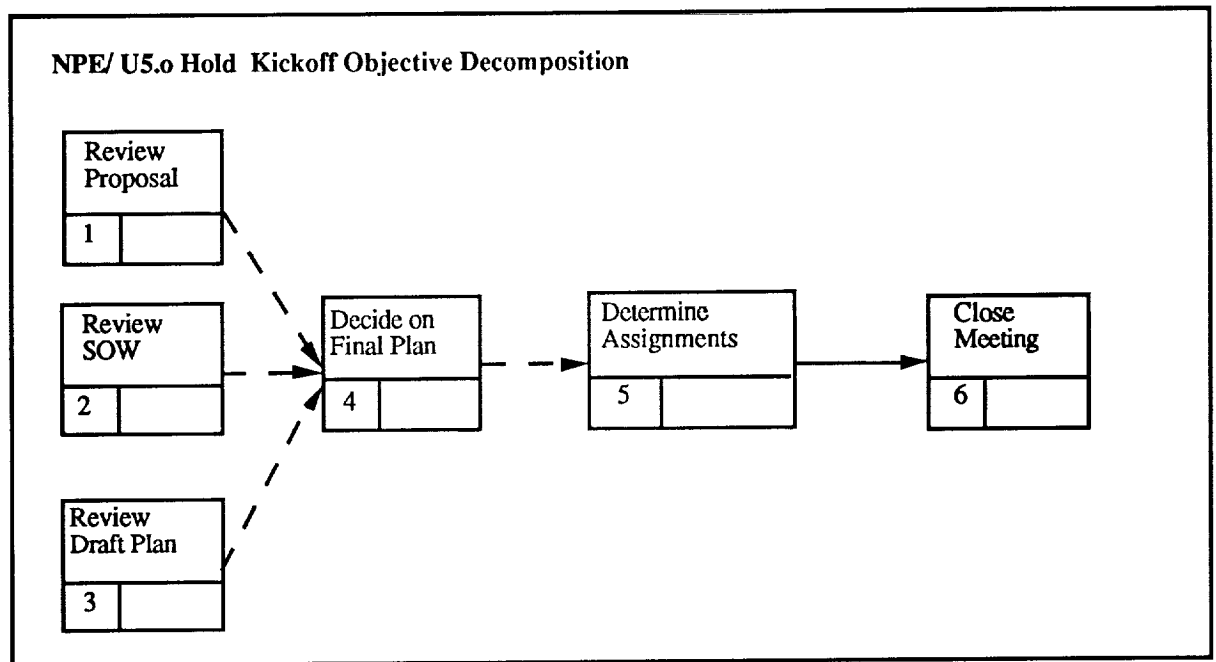


Figure 2.7. Example of an Objective View

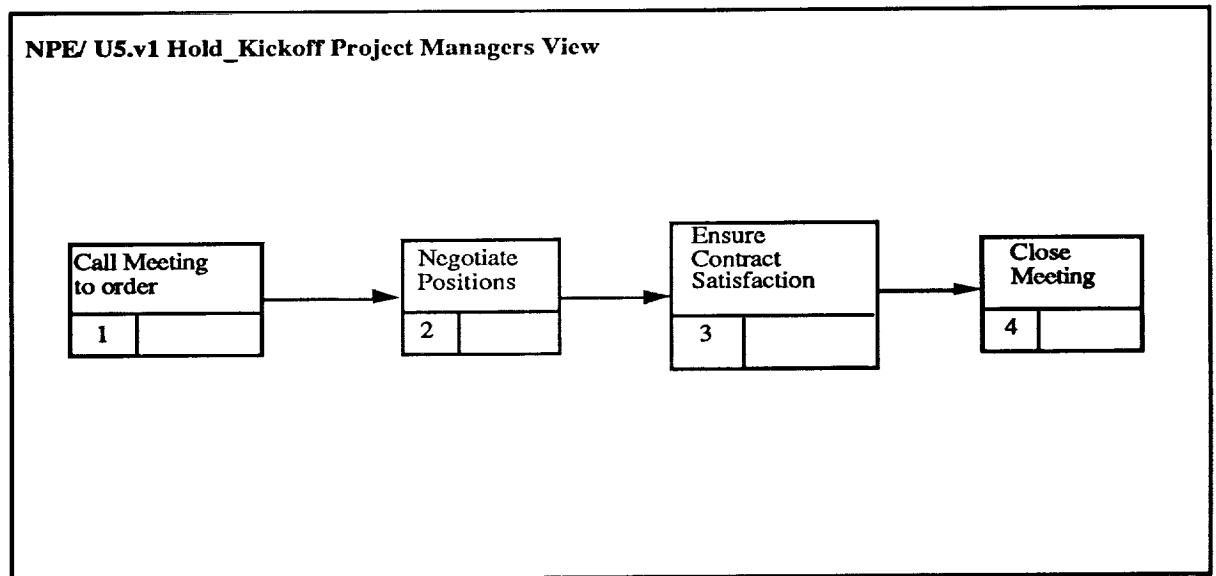


Figure 2.8. Example of a Role View

2.6 Link Types

In IDEF3, links are used to denote distinguished relations between UOBs. The constraint description language within IDEF3 provides a mechanism for describing virtually any type of temporal, logical, conventional, or natural constraint that may exist among a group of UOBs. By 'distinguished', we mean that links are used to highlight important relations that the author of the process description wishes to draw to the attention of the reader. In fact, such relations can be expressed as constraints in the elaboration of the associated UOBs (as described in the previous section). Three default types of links are provided for in IDEF3, 'Dashed Links', 'Precedence Links', and 'Object Flow Links'. These are referred to as 'default types' because in the Model Description Language Section of an IDEF3 description, the author can create new link types with his own default semantics to ease the display of the particular distinguished relations between UOBs that arise in his situation. Unlike IDEF0, there is no significance to the area of connection of a link to a box. Links may start or terminate at any point on a UOB or junction symbols. By convention and wherever possible, process flow descriptions are laid out so that the flow of objects (physical or information) and temporal precedence is ordered from left to right and top to bottom. 'Dashed Links' (DL) carry no predefined semantics. They merely highlight the existence of a relationship between two or more UOBs. This relationship or constraint is specified in the link description. Such a link use allows the author to capture knowledge about a relationship without having to provide a mechanism structure to account for that knowledge.

'Precedence Links' (PL) are a short-hand notation for expression of simple temporal precedence between the instances of one UOB type and those of another UOB type. A PL is denoted by a solid arrow between two UOBs. Informally, the meaning of a precedence link is that each instance of the UOB which is the source of the link completes before the corresponding instance of the destination of the link starts. Precedence links also carry a causality notion in that if two UOBs are connected by a PL then an instance of the first will be followed by an instance of the second, i.e., we can find (or create) a corresponding instantiation of the second UOB to match the first instance.

'Object Flow Links' (OL) are a means of highlighting the participation of the same object instance in two UOB occurrences. An OL carries the same temporal semantics as the PL. An OL is denoted by a solid arrow between a source and a destination UOB with a double arrow head on the destination UOB. It is important to note that lack of an OL link does not imply a change of object participation between two UOBs. It is just the case that the existence of such a link indicates the certain participation of the same object in both UOB occurrences by the author of the diagram.

Each link type has a unique identification number in a model as well as an elaboration (see Figure 2.9) which (like a UOB elaboration) is an collection of expressions in the constraint language. For clarity in the model layout, any link may be deleted by adding the appropriate components of the description associated with the link to the relevant elaborations of the involved UOBs. These descriptions can be extended by the author as the need arises.

Link Specification Form Link Identification Number: _____											
Front UOB Name: _____ Front UOB Label: _____ Front UOB Reference Number: _____ Back UOB Name: _____ Back UOB Label: _____ Back UOB Reference Number: _____											
Object Set: <table style="width: 100%; border: none;"> <tr><td style="border: none;">_____</td><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td><td style="border: none;">_____</td></tr> </table>	_____	_____	_____	_____	_____	_____	_____	_____			
_____	_____										
_____	_____										
_____	_____										
_____	_____										
Fact Set: <table style="width: 100%; border: none;"> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> </table>	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	

Constraint Set: <table style="width: 100%; border: none;"> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> <tr><td style="border: none;">_____</td></tr> </table>	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____	_____

Figure 2.9. Link Specification Form

2.7 Junctions

Junctions in IDEF3 are used to highlight special types of constraints on the possible sequencing relations among UOBs. Junctions in IDEF3 can be used to describe both the logic of a decision making procedure as well as the effect of that logic. This allows the effect of the logic to be displayed at a higher level in the scenario description and the detailed description of the logic to be relegated to a lower level decomposition. Figure 2.10 displays the basic

junction types provided by IDEF3. Junctions in IDEF3 are not UOBs; they do not have a decomposition. In fact, it is best to consider them as macros of the link language. They merely allow commonly used constraints to be expressed quickly and concisely.

There is no commitment to computability in IDEF3 descriptions. That is, one can easily describe a decision procedure for which there is no efficient algorithm as well as situations that are undecidable according to the basic theories of computing. On the other hand, there is a well defined grammar for the use of the junction symbols with the UOB and link types. There is also a well defined semantics for the interpretation of models created within this grammar. If the rules of the syntax are followed, the descriptions produced are guaranteed to have a well formed interpretation, i.e., at least void of any logical inconsistencies. This means that interpretations can be drawn consistently from the descriptions and that in many cases inference and deduction can be performed directly on the models with reliable results.

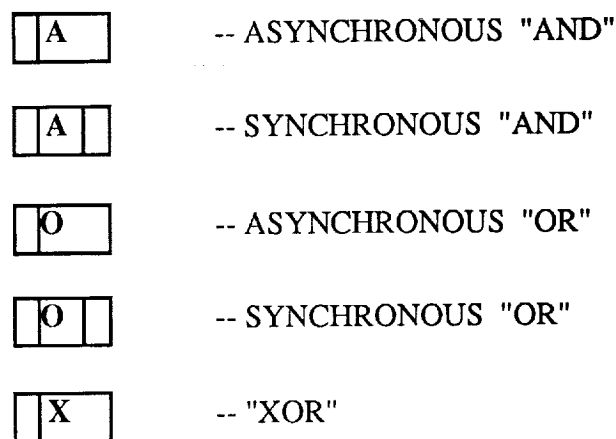


Figure 2.10. Basic IDEF3 Junction Types

There are three major types of junctions 'AND', 'OR', and 'XOR' (exclusive OR). The first two of these junction types come in both a synchronous and an asynchronous interpretation. There is also a different interpretation for the use of junctions to initiate a branching and junctions used to terminate a number of links (fan-out interpretation versus fan-in interpretation). Figures 2.11 and 2.12 provide a summarization of the definitions for each junction type both at the front and at the back of a link set. Several points should be noted from these diagrams. First, the semantics of these junctions includes elements of both logical operators and instantiation control. That is, when the conditions of a junction are satisfied, the following linked UOB(s) will be enabled for instantiation. For this reason the front and the back of link set semantics for junctions are not strictly duals of each other as can be seen from

Figures 2.11 and 2.12. For instance an asynchronous 'OR' junction at the front of a link set allows for any combination of the attached UOBs to happen in any order in time. However, such a junction at the back of a link set only allows for a single ending UOB or a set of UOBs that complete together.

Note that there is no provision for synchronous or asynchronous 'XOR' as this junction provides only for one of the following UOBs to be instantiated, obviating the possibility or need for any type of synchronization.





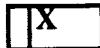
JUNCTION TYPE		MEANING
	-- ASYNCHRONOUS "AND"	ALL FOLLOWING UOBs WILL EVENTUALLY HAPPEN
	-- SYNCHRONOUS "AND"	ALL FOLLOWING UOBs WILL HAPPEN AND START TOGETHER
	-- ASYNCHRONOUS "OR"	1, OR MANY OF THE FOLLOWING UOB's WILL EVENTUALLY HAPPEN
	-- SYNCHRONOUS "OR"	THE START OF WHATEVER COMBINATION OCCURS IS SYNC'd
	-- "XOR"	EXACTLY ONE OF THE FOLLOWING UOB's WILL EVENTUALLY HAPPEN

Figure 2.11. Front of Link Set Semantics For Junction Types

Finally, there are special interpretations for the use of combinations of junctions, e.g., an AND with a following OR, versus an OR with a following AND. In the remainder of this section, we will describe each of the types of junctions and their various uses/meanings as well as the process for interpreting sets of combined or mixed uses of these symbols. In our experience to date, it so happens that most uses of junctions occur in pairs. This is not to say that IDEF3 prevents the use of a junction to indicate the initiation of multiple processes that proceed independently, quite the contrary. However in the descriptions of most organizational scenarios and corresponding user roles or information support transactions, the parallel activities almost always converge.






JUNCTION TYPE		MEANING
	-- ASYNCHRONOUS "AND"	ALL PRECEDING ATTACHED UOBs MUST HAVE COMPLETED
	-- SYNCHRONOUS "AND"	ALL PRECEDING ATTACHED UOBs MUST COMPLETE TOGETHER
	-- ASYNCHRONOUS "OR"	1, OR MORE OF THE PRECEDING UOB's HAS COMPLETED
	-- SYNCHRONOUS "OR"	THE COMPLETION OF WHATEVER COMBINATION OCCURRED WAS SYNC'd
	-- "XOR"	EXACTLY ONE OF THE PRECEDING UOB's HAS HAPPENED

Figure 2.12. Back of Link Set Semantics For Junction Types

Figure 2.13 illustrates one of the most frequently used junction types (the asynchronous 'AND' junction). In the scenario depicted in this figure, the completion of the receipt of a proposal is followed by both a cost and a technical evaluation which must both be completed prior to the award of the contract. Note that there is no specified timing relationship between the cost and technical evaluation but that both must follow the receipt, and both must precede the award.

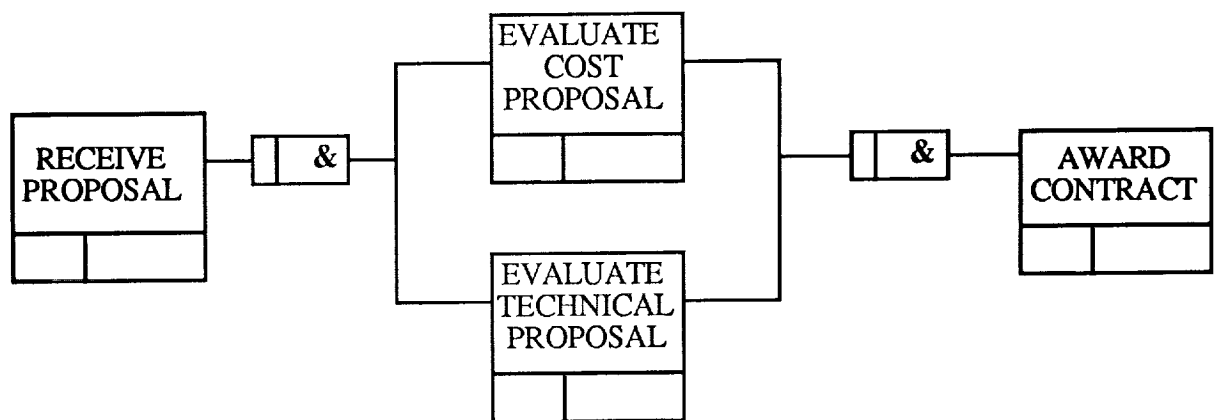


Figure 2.13. Asynchronous "AND" Junction Example

Contrast this to the scenario displayed in Figure 2.14 in which the synchronous 'AND' describes a situation in which the cost and technical evaluation must start simultaneously, but still may end separately. Had there been an organizational rule that required both these to end together as well,

there would have been a synchronous 'AND' at the terminal end of the branch as well.

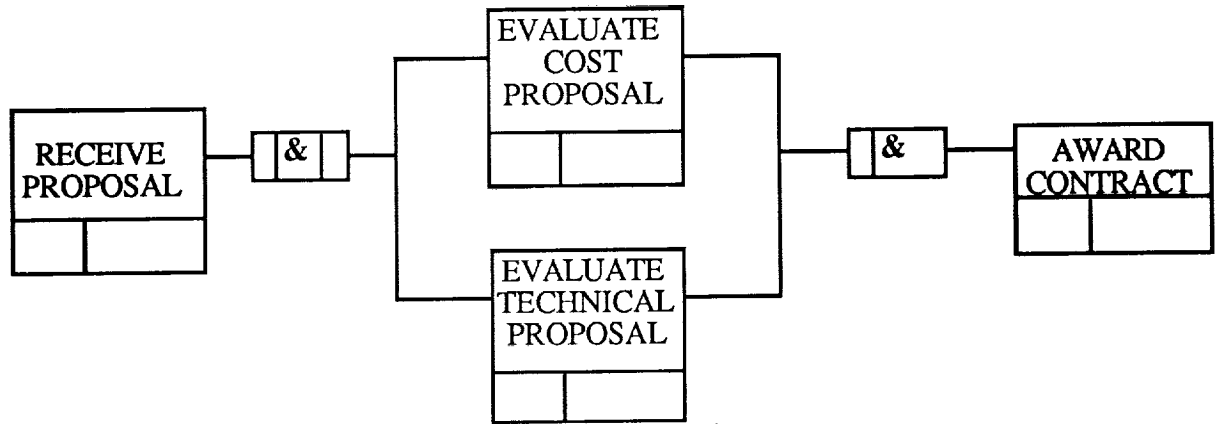


Figure 2.14. Synchronous "AND" Junction Example

In Figure 2.15, we see a description of the proposal evaluation process, hypothetically one which could have been a decomposition of either the cost or the technical evaluation processes described in Figures 2.13 and 2.14. This process description states that following the evaluation, one can reject the proposal, negotiate changes, or accept the proposal or some combination of these actions.

In this scenario description, the rejection of a proposal is a terminating activity; however, either of the other two activities (or both) can result in the enablement of a contract award. An important point to note in this description is the obvious lack of specifics relative to possible interactions between the negotiation of changes to a proposal and the proposal acceptance activity. In some situations, this may be a complete description in that the original proposal is either accepted or not. If not, contract award is predicated on the contractor's accepting the terms of the funding agency. In other situations, the contractor may be asked to resubmit proposals as a part of the negotiation process. Either such situation can be easily represented in IDEF3.

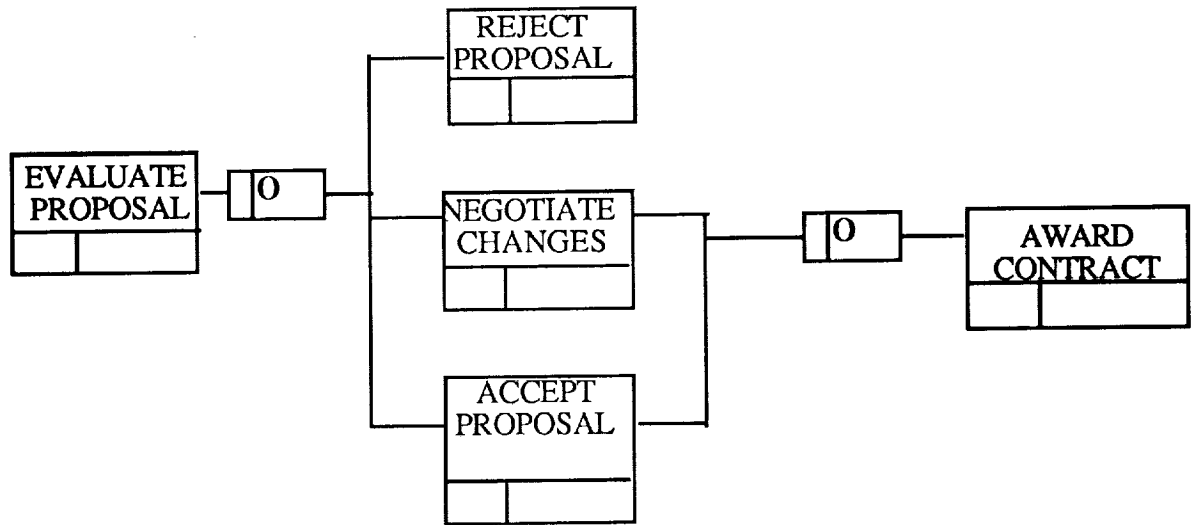


Figure 2.15. Asynchronous "OR" Junction Example

The final example in this section is of a pathological description which should never be formulated in IDEF3. Figure 2.16 provides this example. In that figure, a situation is described for which there is no possible realization. Since the XOR describes a situation where only one of the following processes can be realized the AND junction conditions can never be satisfied. While in a small example, the inconsistency of this situation is obvious, it is quite often the case that such inconsistencies really do exist in the domain being modeled. This is usually the result of evolved policies originating in different organizations at different points in time by different individuals, probably responding to different pressures. Part of the purpose in the use of a description capture method like IDEF3 is to raise these inconsistencies within the organization to allow their resolution.

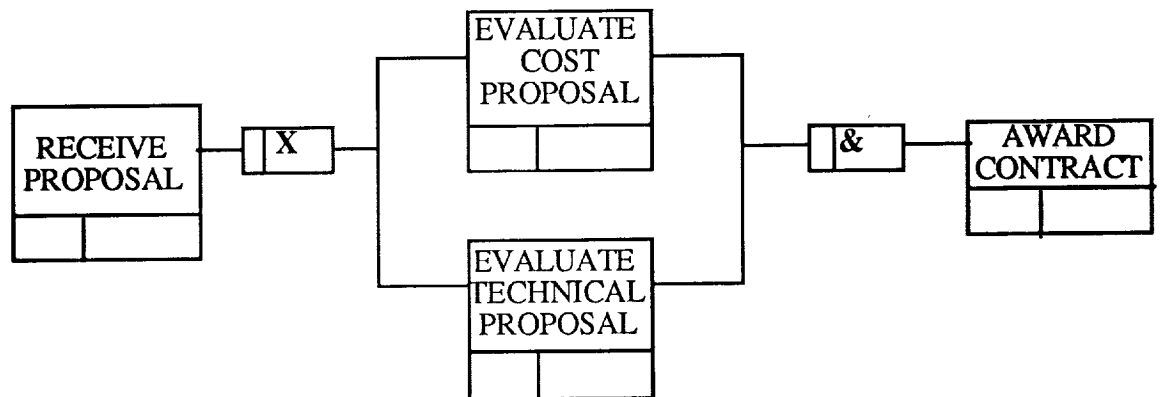


Figure 2.16. Pathological "XOR"/"AND" Junction Combination example

Elaborations may be attached to a junction through the use of the 'Reference Connector' notation described in the next section. This feature allows the author to record information about decision logic associated with a junction. For example, in the case of an OR junction, the author may have knowledge of the conditions under which the individual branches are chosen. Such knowledge is expressed in an elaboration form that would be attached to the particular junction via a reference connector.

2.8 Reference Connectors

Reference connectors allow the IDEF3 author to:

- 1) Span multiple pages in a diagram layout,
- 2) Indicate a loop back to a previously defined UOB,
- 3) Indicate that another instance of a previously defined UOB is to be inserted at a specific point in the process (without loop back),
- 4) Emphasize the description of particular objects or relations in the elaboration,
- 5) Tie in specific examples of referenced data or objects (e.g., screen layouts),
- 6) Associate special constraint sets to junctions (indicate that a junction has an elaboration that contains the facts, constraints or decision logic which describe how that junction works),
- 7) Form references between the Process Flow Descriptions and the Object State Transition descriptions.

It is often the case that users of IDEF3 will find the reference connectors a flexible way to express ideas or concepts in lieu of the junction types and dashed arrows/constraint language statements. The basic symbol structure of the reference connectors is displayed in Figure 2.17. Though not shown in this figure, referents to UOBs may carry a synchronous or asynchronous designator identical to that of the junction symbols described previously.

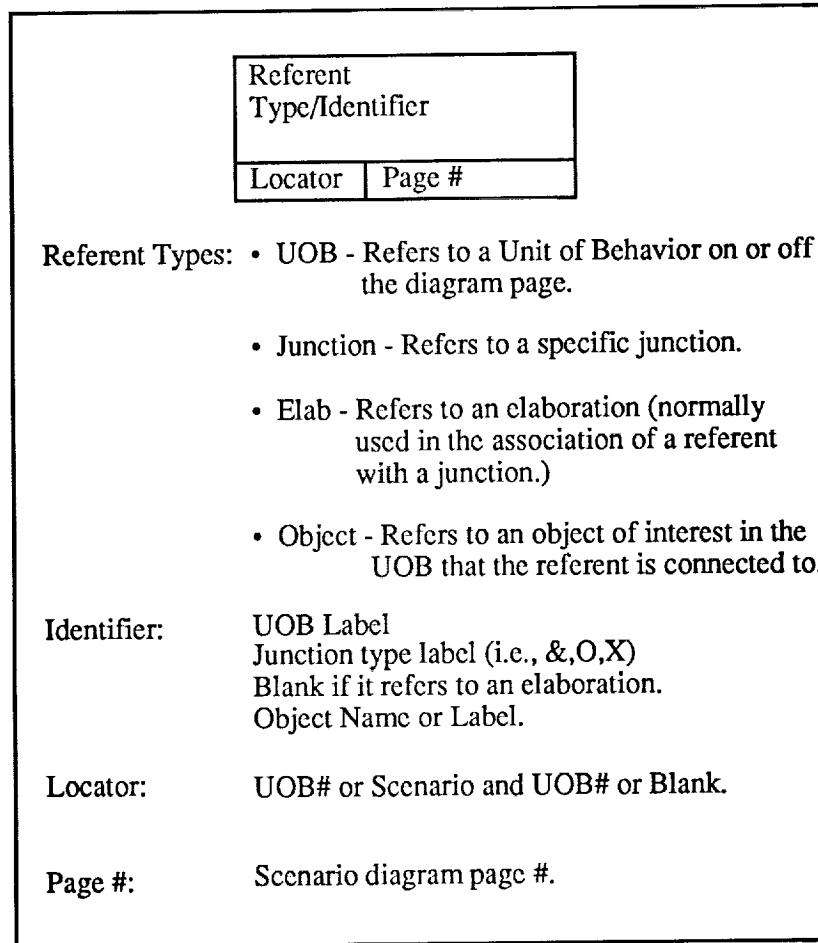


Figure 2.17. Referent Syntax.

2.9 Object State Transition Description

The object state transition diagram was included in the IDEF3 method to allow for the construction of an object centered view of the organization or system. Such a view cuts across the process diagrams and summarizes the allowable transitions of an object in the domain. Object centered views have been found to be useful particularly in the documentation of data life cycles. IDEF3 object state transition descriptions allow the characterization of the states that an object is known to exist in. This characterization includes 1) the conditions for classification in a state, 2) the properties of objects in a state, and 3) the conditions for transition out of that state.

Descriptions of the states of an object and the allowable transitions between such states are collected together and displayed on an object state transition network diagram. The basic structure of an object state transition network (OSTN) diagram in IDEF3 is modeled after Augmented Transition Networks (ATNs). ATNs are capable of representing any computable function [Aho 78].

Figures 2.18 and 2.19 display the basic elements of the object state transition network diagram. The nodes (solid boxes) in the network represent object states. Each OSTN has an associated elaboration form. Similarly, each object state in an OSTN also has an elaboration form. The contents of these elaboration forms will be described later in this section. Reference connectors attached to the arcs of an object state transition network are used to refer to other object transition networks, UOBs, or Scenarios that affect the transition of an object from one state to the next. The meaning of an arc labeled with an OSTN reference is that the corresponding named OSTN must be completed before a transition across that arc can be completed. Similarly, if an arc is labeled with a referent to a UOB or Scenario, this indicates the dependence of that transition on instances of the UOB or Scenario. For example, in Figure 2.18 the UOB *Perform Concept Demonstration & Validation* must be initiated and completed before a system can transition from Milestone 1 to Milestone 2. In Figure 2.19 the asynchronous referent to a scenario indicates that that an instance of that scenario must be initiated but need not complete prior to the transition of the object from state 1 to state 2.

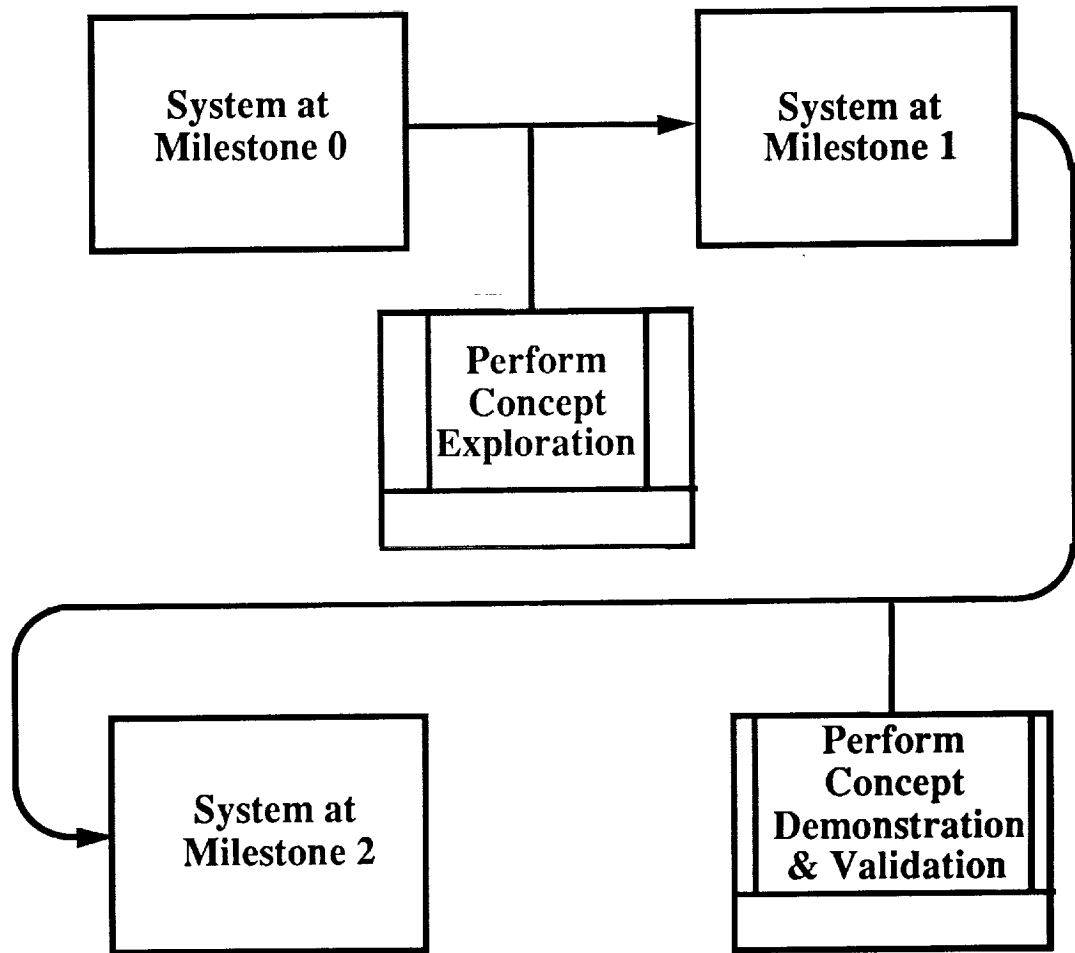


Figure 2.18. Object State Transition Network Example

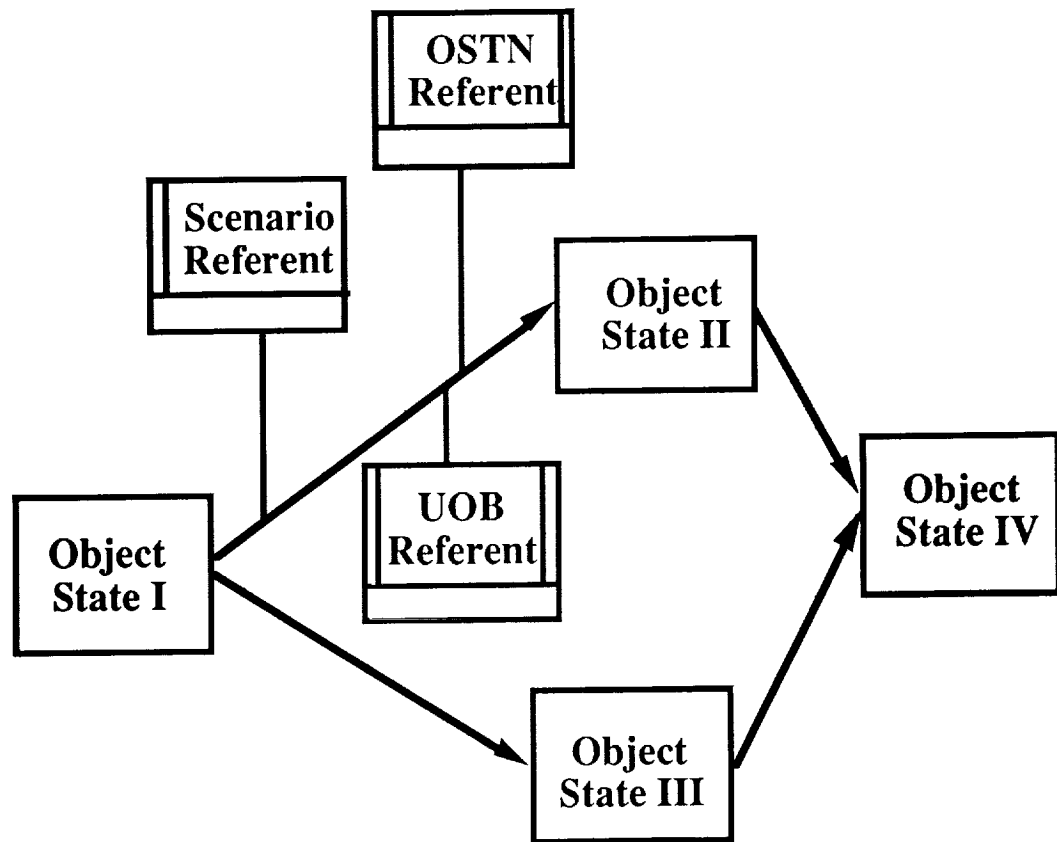


Figure 2.19. Object State Transition Network Example

Though not displayed in Figure 2.18, each model element in this component of the IDEF3 description carries the following attributes of (1) unique identifier, (2) label (displayed with the symbol), and (3) description (an elaboration). The elements of an OSTN elaboration that are captured on the Object State Transition Network Description Form displayed in Figure 2.20 include:

- 1) OSTN Name: Display label of the object state transition network (OSTN)
- 2) OSTN #: Unique identifier of the OSTN.
- 3) Object Name: Name of object that is the focus of this OSTN
- 4) OSRN Glossary: Textual Description of object state transition network
- 5) OS Set: Set of object states that make up the OSTN
- 5) Scenario Set: Names of scenarios referenced in the OSTN
- 6) UOB Set: Names of UOBs referenced in this OSTN
- 7) OSTN Set: Names of OSTNs referenced in this OSTN

IDEF 3 Object State Transition Network Description Form	
OSTN Name: _____ OSTN #: _____	
Object Name: _____	
OSTN Glossary: _____ _____ _____ _____ _____ _____	
OS Set: _____ _____ _____ _____	
Scenario Set: _____ _____ _____ _____	UOB Set: _____ _____ _____ _____
OSTN Set: _____ _____ _____ _____	_____ _____ _____ _____

Figure 2.20. IDEF3 Object State Transition Network Form

The elements of an object state elaboration that are captured on the Object State Description Form displayed in Figure 2.20 include:

- 1) Object State Name: Full unique name of the object state,
- 2) Object State Label: Label which appears on object state diagrams,
- 3) Object State #: Unique number for the object state,

- 4) OSTN Set: Names of the Object State Transition Networks to which the object state belongs,
- 5) Post-Transition Restrictions: Properties, Facts, or constraints which must hold prior to admission of an object into this state,
- 6) State Description: Properties, Facts, or constraints which must hold for all objects in the state,
- 7) Pre-Transition Restrictions: Properties, Facts, or constraints which must hold prior to initiating a state transition attempt,
- 8) Glossary: Textual descriptions of the object state.

IDEF 3 Object State Description Form	
OS Name: _____ Object Name: _____ OS Label: _____ OS Reference Number: _____	
OSTN Set: _____ _____ _____ _____	Glossary: _____ _____ _____ _____
Post Transition Restrictions: Properties: _____ _____ Facts: _____ _____ Constraints: _____ _____	
State Description: Properties: _____ _____ Facts: _____ _____ Constraints: _____ _____	
Pre Transition Restrictions: Properties: _____ _____ Facts: _____ _____ Constraints: _____ _____	

Figure 2.21. IDEF3 Object State Description Capture Form

It is important to note that the features that define an object state may or may not be attributes currently known to the information system. If they are, then the OS elaboration allows for the definition of cross references to the IDEF1 model entities or attributes. In the pre- and post- transition restrictions, conditions in terms of value restrictions on the features are normally expressed. Most of the time, simple value restrictions on attributes of the object will suffice for these transition restrictions. If simple value restrictions are not convenient or powerful enough, then a set of ISyCL constraint statements may be used. Note that the 'Post Transition Restrictions' are those constraints that must be satisfied before a transition into a state is allowed. Note that the 'Pre-Transition Restrictions' are those constraints that must be satisfied before a transition into a state can be attempted. Notice also that the Pre and Post constraints may contain references to attributes that are not defining attributes of the object state.

3.0 IDEF3 Description Formulation

At the time of this report, the IDEF3 method is still in beta testing. The best method for building a process flow and object state description appears to be sensitive to the intended use of that description. In this section we summarize several tailored methods that are being experimented with for the purpose of constructing IDEF3 descriptions.

The basic approach for building an IDEF3 description for the purpose of documentation of the user view of the system requirements can be summarized in the following steps:

- 1) Identify scenarios.
 For each Scenario do:
 Identify agent objects, recipient objects, focus objects.
 Identify participating UOBs in that scenario
- 2) For each object, identify states and state transitions.
 Identify states (situations), events, actions, processes.
- 3) Identify process sequence for processes which account for focus object transitions.
- 4) Construct process flow charts.
- 5) Construct OSTNs.
- 6) Check for consistency and completeness in the descriptions.
- 7) Validate description with domain experts.

If the purpose is to use the Process Flow Model for User Role Driven Design (i.e., as a design specification to the software developer), then the following activities must be added:

- 1) Treat each process in process flow diagram as a scenario.
- 2) Identify information to be displayed to the user.
- 3) Identify actions which user can perform.
- 4) Determine response of system to each action user can perform.
- 5) Identify process sequence of allowable user actions.
- 6) Determine information system state change as a result of each process in the sequence.
- 7) Package information to be displayed to the user at each step in the process sequence.

IDEF3 is similar to IDEF0 in its sensitivity to establishing a viewpoint. If the purpose is to use the Process Flow Model for User Role Driven Design then the following three view types are recommended in the application of the above described method:

- The environment system (organization) view.
- The client (or user) view.
- The agent (e.g. target information system) view.

Within an information resource management application arena, these general views will normally translate as follows:

The **environment system** view will most often correspond to an organizational system view.

The **client** view will normally correspond to the user functions.

The **agent** view will normally correspond to the information system response to the user functions. This view will normally include the target system view which corresponds to an information system view.

4.0 Summary

This report has presented an overview of the foundations and content of the evolving IDEF3 process flow and object state description capture method. This method is currently in beta test. Ongoing efforts in the formulation of

formal semantics models for descriptions captured in the outlined form and in the actual application of this method can be expected to cause an evolution in the method language.

5.0 Bibliography

- Allen J. F., "Towards a General Theory of Action and Time," *Artificial Intelligence* (23), 1984, pp 123-154.
- Balci O., Nance R. E., "Simulation Model Development Environments: A Research Prototype," *Journal of the Operational Research Society*, 38(8), 1987, pp 753-763.
- Barwise, J. and Perry, J., *Situations and Attitudes*, The MIT Press, Cambridge, 1983.
- Cross S. , "Qualitative Reasoning in an Expert System Framework", T-124 University of Illinois Coordinated Science Lab, Urbana, IL, 1983.
- De Kleer J., "Causal and Teleological Reasoning in Circuit Recognition," TR-529, MIT AI Lab, Cambridge, MA, 1979.
- De Kleer J., Brown J. S., "A Qualitative Physics Based on Confluences," *Artificial Intelligence* (24), 1984, pp 7-83.
- Devlin, K., *Logic and Information, Volume I: Situation Theory*, Cambridge University Press.
- Forbus K. , "Qualitative Process Theory," *Artificial Intelligence* (24), pp 85-168, 1984.
- KBSI 91a, "Description Debugging with Qualitative Reasoning," Technical Report, Knowledge Based Systems, Inc., August 1991.
- KBSI 91b, "KBSA Concept of Operations Document," Technical Report, Knowledge Based Systems, Inc., August 1991.
- KBSI 91c, "Qualitative Reasoning in Model Design," Technical Report, Knowledge Based Systems, Inc., August 1991.
- KBSI 91d, "KBSA System Architecture Document," Technical Report, Knowledge Based Systems, Inc., August 1991.
- KBSI 91e, "Knowledge Engineering Results for Simulation Modeling," Technical Report, Knowledge Based Systems, Inc., August 1991.
- KBSI 91g, "Qualitative Reasoning for Problem Solving from IDEF3 Descriptions," Technical Report, Knowledge Based Systems, Inc., August 1991.
- Kripke, S., "Semantical Considerations on Modal Logic," *Acta Philosophica Fennica* (16), 1963, pp 39-48.

- Lin, Min-Jin, "Automatic Simulation Model Design from a Situation Theory Based Manufacturing System Description," PhD Dissertation, Texas A & M University, May 1990.
- Link, G., "The Logical Analysis of Plurals and Mass Terms: A Lattice Theoretic Approach," in R. Bauerle *et al.* (eds), *Meaning, Use, and Interpretation*, Berlin, De Gruyter, 1983.
- Mayer R. J., "Cognitive Skills in Modeling and Simulation," PhD Dissertation, Texas A&M University, December 1988.
- Mayer, R.J., Cullinane, T., Knappenberger, B., Wells, S., "IDEF3 Practice and Use Technical Report," Interim Technical Report, Armstrong Laboratory, Logistics Research Division, Wright-Patterson Air Force Base, OH Aug 1991b.
- Menzel, C., "Actualism, Ontological Commitment, and Possible World Semantics," *Synthese* 85 (1990), 355-389
- Menzel C. P., Mayer R. J., Edwards D. D., "IDEF3 Formalization Report," Final Technical Report, IISEE Project, Armstrong Laboratory, Logistics Research Division, Wright-Patterson Air Force Base, OH 1991.
- Menzel, C.P., Mayer, R. J., and Edwards, D., "IDEF3 Process Descriptions and Their Semantics," forthcoming in Kuziak, A., and Dagli, C., *Knowledge Base Systems in Design and Manufacturing*, Chapman Publishing, forthcoming 1991.
- Menzel, C., and R. Mayer, "Theoretical Foundations for Information Representation and Constraint Specification," final technical report, IISEE project, AFHRL/LRL, WPAFB, Ohio, March 1991.
- Overstreet C. M., Nance R. E., "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Communications of the ACM*, 28(2), 1985, pp 190-201.
- Pegden C. D., *Introduction to SIMAN*, Systems Modeling Corp., 1982.
- Pritsker A. A.B., Pegden C. D., *Introduction to Simulation and SLAM*, Halsted Press, New York, 1979.
- Poole D., "A Methodology for Using a Default and Abductive Reasoning System," *International Journal of Intelligent Systems*, 5, 1990, pp 521-548.
- Sanders L., Mayer R. J., Menzel C. P., Keen A. A., Browne D., "Description Representation with Container objects," *Conference Proceedings of Autofact 91*, 1991.
- Shannon R. E., Mayer R. J., Adelsberger H., "Expert Systems and Simulation," *Simulation* 44(6), 1985, pp 275-284.
- Simmons R. , "Representing and Reasoning about Change in Geologic Interpretation," TR-749, MIT AI Lab, Cambridge, MA, 1983.

Tarski, A., "The Concept of Truth in Formalized Languages," *Logic, Semantics, and Metamathematics*, Oxford University Press, Oxford, 1956.

