AL-TR-1995-XXXX

INFORMATION INTEGRATION FOR CONCURRENT ENGINEERING (IICE) IDEF3 PROCESS DESCRIPTION CAPTURE METHOD REPORT

Richard J. Mayer, Ph.D.
Christopher P. Menzel, Ph.D.
Michael K. Painter
Paula S. deWitte, Ph.D.
Thomas Blinn
Benjamin Perakath, Ph.D.

KNOWLEDGE BASED SYSTEMS, INCORPORATED
ONE KBSI PLACE
1500 UNIVERSITY DRIVE EAST
COLLEGE STATION, TEXAS 77840-2335

HUMAN RESOURCES DIRECTORATE
LOGISTICS RESEARCH DIVISION

SEPTEMBER 1995

INTERIM TECHNICAL REPORT FOR PERIOD APRIL 1992 - SEPTEMBER 1995

September 1995
Interim - February 1991 to September 1995
IDEF3 Process Description Capture Method Report

Richard J. Mayer, Ph.D. Michael K. Painter
Christopher P. Menzel, Ph.D. Benjamin Perakath, Ph.D
Paula S. deWitte, Ph.D. Thomas Blinn

Knowledge Based Systems, Inc.
One KBSI Place, 1500 University Drive East
College Station, TX 77845

KBSI-IICE-90-STR-01-0592-02

Armstrong Laboratory
Human Resources Directorate
Logistics Research Division
Wright-Patterson AFB, OH 45433

AL-TR-1995-XXXX
Approved for Public Release; distribution is unlimited
A

This document provides a method overview, practice and use description, and language reference for the Integration Definition (IDEF) method for Process Description Capture (IDEF3). IDEF3 is designed to help document and analyze the processes of an existing or proposed system. Proven guidelines and a language for information capture help users capture and organize process information for multiple downstream uses. IDEF3 supports both process-centered and object-centered knowledge acquisition strategies enabling users to capture assertions about real-world processes and events in ways paralleling common forms of human expression. IDEF3 includes the ability to capture and structure descriptions of how a system works from multiple viewpoints. As an integral member of the IDEF family of methods, IDEF3 works well in independent application or in concert with other IDEF methods to document, analyze, and improve the vital processes of a business.

process, IDEF, method, methodology, modeling, knowledge acquisition, requirements definition, information systems, information engineering, systems engineering, integration, reengineering

Unclassified
Unclassified
Unclassified
UL
C F33615-90-C-0012
PE 63106F
PR 2940
TA 01
WU 08
140

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

This document provides a method overview, practice and use description, and language reference for the IDEF3 Process Description Capture Method developed under the Information Integration for Concurrent Engineering (IICE) project, F33615-90-C-0012, funded by Armstrong Laboratory, Logistics Research Division, Wright-Patterson Air Force Base, Ohio 45433, under the technical direction of United States Air Force Captain JoAnn Sartor and Mr. James McManus. The prime contractor for IICE is Knowledge Based Systems, Inc. (KBSI), College Station, Texas. Dr. Paula S. deWitte was the IICE Project Manager at KBSI. Dr. Richard J. Mayer was the Principal Investigator on this project. Mr. Thomas Blinn was the IICE Technical Manager and also served as the Project Manager during the close out of this effort. Michael K. Painter was the Methods Engineering thrust manager. The authors gratefully acknowledge the technical contributions of the following people.

Bruce E. Caraway
Thomas P. Cullinane, Ph.D.
John W. Crump, IV
Florence Fillion
Mike Graul, Ph.D.
Richard Henderson
Arthur Keen, Ph.D.
William K. Knappenberger
Madhavi Lingineni
M. Sue Wells

KBSI also acknowledges the technical input to this document made by previous work under the Integrated Information Systems Evolutionary Environment (IISEE) project sponsored by the Armstrong Laboratory Logistics Research Division and performed by the Knowledge-Based Systems Laboratory, Department of Industrial Engineering, Texas A&M University.

FOREWORD

Significant technological, economic, and strategic benefits can be attained through the effective capture, control, and management of information and knowledge resources. Like manpower, materials, and machines, information and knowledge assets are recognized as vital resources that can be leveraged to achieve a competitive advantage. The Air Force Information Integration for Concurrent Engineering (IICE) program, sponsored by the Armstrong Laboratory's Logistic Research Division, was established as part of a commitment to further the development of technologies that will enable full use of these resources.

The IICE program was chartered with developing the theoretical foundations, methods, and tools to successfully evolve toward an information-integrated enterprise. These technologies are designed to leverage information and knowledge resources as the key enablers for high quality systems that achieve better performance in terms of life cycle cost and efficiency. The methods research described in this report reflects recent advancements in technology for leveraging available information and knowledge assets.

The name IDEF originates from the Air Force program for Integrated Computer-Aided Manufacturing (ICAM) which developed the first ICAM Definition, or IDEF, methods. Continued development of IDEF technology supports an overall strategy to provide a family of mutually-supportive methods for enterprise integration.. More recently, with the expanded focus and use of IDEF methods as part of Concurrent Engineering, Total Quality Management (TQM), and business re-engineering initiatives, the IDEF acronym has been re-cast as an integrated family of Integration Definition methods. Before discussing the development strategy for providing an integrated family of IDEF methods, the components and general nature of methods are described.

Method Anatomy

A method is an organized, single-purpose discipline or practice (Coleman, 1989). A method may have a formal theoretical foundation, although this is not a requirement. Generally, methods evolve as a distillation of the *best-practice* experience in a particular domain of activity. The term tool is used to refer to a software system that supports the application of a method.

Though a method may be thought of informally as a procedure for doing something plus (perhaps) a representational notation, it may be described more formally as consisting of three components (Figure 1). Each method has (1) a definition, (2) a discipline, and (3) a use component. The definition specifies the basic intuitions and

motivations behind the method, the concepts involved, and the theory of its operation. The discipline component includes the syntax of the method and the procedure for applying the method. The method procedure gives the practitioner a reliable process for consistently achieving good results. The method syntax eliminates ambiguity when developing complex engineering products. Many system analysis and engineering methods use a graphical syntax to provide visualization of collected data so that key information can be easily extracted.[1] The use component characterizes how to successfully apply the method in different situations.

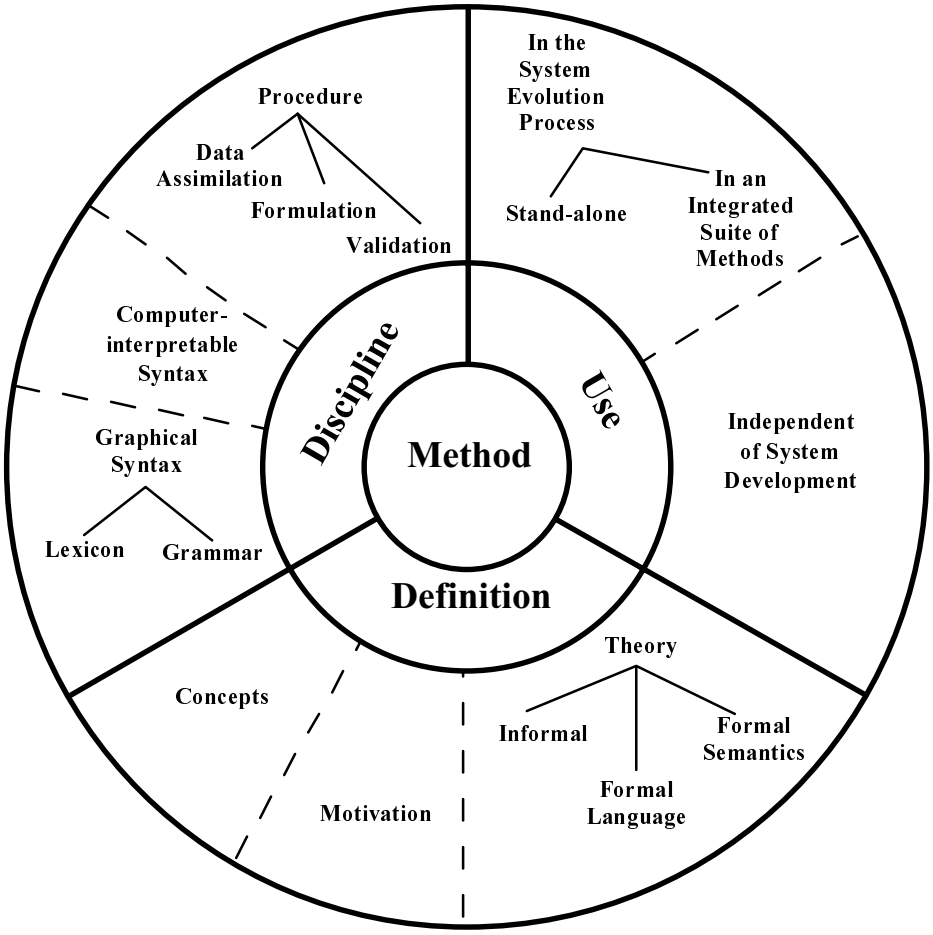

Figure 1-1
Anatomy of a Method

[1] Graphical facilities provided by a method language serve not only to document the analysis or design process undertaken, but more importantly, to highlight important decisions or relationships that must be considered during method application. The uniformities to which an expert, through experience, has become attuned are thus formally encoded in visualizations that emulate expert sensitivities.

Ultimately, methods are designed to facilitate a scientific approach to problem solving. This goal is accomplished by first creating an understanding of the important objects, relations, and constraints that must be discovered, considered, or determined. Second, scientific problem solving occurs by guiding the method practitioner through a disciplined approach that is consistent with good-practice experience and leads toward the desired result. Formal methods, then, are specifically designed to raise the performance level (quality and productivity) of the novice practitioner to a level comparable to that of an expert (Mayer, 1987).

Family of Methods

John Zachman, in his pioneering work on information systems architecture, observed:

> [T]here is not an architecture, but a set of architectural representations. One is not right and another wrong. The architectures are different. They are additive, complementary. There are reasons for electing to expend the resources for developing each architectural representation. And, there are risks associated with not developing any one of the architectural representations.

The consistent, reliable creation of correct architectural representations requires the use of a guiding method. These observations underscore the need for *many* «architectural representations,» and, correspondingly, many methods.

Methods, and their associated architectural representations, focus on a limited set of system characteristics and ignore those that do not pertain to the task at hand. Methods are not intended to evaluate and represent every possible state or characteristic of the system under study. Hence, the search for a single method, or modeling language, supporting the specification, analysis, design, and representation of all relevant system characteristics continues to frustrate those making the attempt. If such a goal were achievable, the exercise would itself build the actual system, negating the benefits of method application (e.g., problem simplification, low cost, rapid evaluation of anticipated performance, and so forth).

On the other hand, lack of integration among special-purpose methods can be equally frustrating. The IDEF family of methods is intended to strike a balance between special-purpose methods, which are limited to specific problem types, and «super methods» which attempt to include everything. This balance is maintained by providing explicit mechanisms for integrating the results of individual methods within the IDEF family.

Previous research identified critical needs for new methods[2] and led to a renewed effort in IDEF method development, with a mandate for compatibility among the family of IDEF methods. New method development has gone in directions where obvious voids existed (rather than reinventing existing methods) with the mission to forge integration links among existing IDEF methods. When applied in a stand-alone fashion, IDEF methods embody knowledge of good practice for the targeted activity. As with any good method, the IDEF methods are designed to raise the performance level of novice practitioners by focusing attention on important decisions while masking irrelevant information and unneeded complexity. Viewed as a toolbox of complementary methods technology, the IDEF family is designed to promote the integration of effort in an environment where effective results have become increasingly dependent on effective use of enterprise information and knowledge assets.

[2]Notably, the Knowledge-Based Integrated Information Systems Engineering Project was conducted at the Massachusetts Institute of Technology (MIT) in 1987, where a collection of highly qualified experts from academic and research organizations, government agencies, computer companies, and other corporations identified method and tool needs for large-scale, heterogeneous, distributed systems integration. See Defense Technical Information Center (DTIC) reports A195851 and A195857.

INTRODUCTION

One of the most common communication mechanisms to describe a situation or process is a story told as an ordered sequence of events or activities. For example, an engineer often describes the design process of his company by telling a story about a product that was recently developed. Likewise, a shop floor supervisor may describe the operation of his manufacturing system by describing the process of building a product in his shop.

The IDEF3 Process Description Capture Method was created specifically to capture descriptions of sequences of activities. The primary goal of IDEF3 is to provide a structured method by which a domain expert can express knowledge about the operation of a particular system or organization. Knowledge acquisition is enabled by direct capture of assertions about real-world processes and events in a form that is most natural for capture. This includes the capture of assertions about the objects that participate in the process, assertions about supporting objects, and the precedence and causality relations between processes and events in the environment.

IDEF3 supports this kind of knowledge acquisition by providing a reliable and well-structured approach for process knowledge acquisition, and an expressively powerful, yet easy-to-use, language for information capture and expression. These two dimensions of IDEF3—the procedure embodying proven practices and an expressively powerful language—work together to focus user attention on relevant aspects of a given process and provide the expressive power necessary to explicitly represent information about the nature and structure of that process.

Motivation

In the most general sense, a process issimply an ordered sequence of events. In human-designed systems, the events that constitute a process are designed and ordered to achieve some desired outcome. A business process, in particular, is an ordered sequence of events involving people, materials, energy, and equipment that is designed to achieve a defined business outcome [(Davenport & Short, 1993, 103), (Pall, 1987)]. The importance of business processes is self-evident. They not only define what the business does, but more importantly, they determine how well the business does what it does.

Several motivating factors led to the development of IDEF3. Some of the more prominent motivations are described in the following sections.

Enhance the Productivity of Business Systems Analysis

One major motivation behind IDEF3 development was the need to speed up the process of business systems modeling. Business systems analysis often begins by acquiring an accurate description of the problem situation. Domain experts express recurring situations in terms of an ordered sequence of events or activities. Moreover, domain experts generally describe the specific ways in which activities and the objects that participate in them are related. There is a need for both a *method* to facilitate the capture of the dynamics of business activities and process descriptions, and for a *representation medium* to store and manipulate this captured knowledge. IDEF3 fulfills these requirements with a structured approach to communicate such process information described by domain experts.

Facilitate Design Data Life-Cycle Management

Earlier studies to identify method needs revealed the lack of methods to capture descriptions of design-data life cycles (Mayer 1987). To describe the engineering design-data life cycle, it is necessary to describe: (1) design information artifacts (i.e., drawings, CAD models, etc.), (2) state transitions through which these artifacts proceed, and (3) the decision logic or processes that determine the state transitions. IDEF3 provides mechanisms to describe this data life cycle information.

Support the Project Management Process

Project management techniques are used to monitor and control projects in various application domains. Several software tools support these project management techniques. However, many of these techniques are not expressively powerful enough to capture many of the complexities that occur in project management situations. IDEF3 provides mechanisms to capture the constraints (including resource and temporal relationships) between the activities of a project. The IDEF3 language also represents detailed information about the objects that participate in, are produced by, or are used by the project activities. Furthermore, the activation of IDEF3 diagrams, which can be supported by an automated tool, will provide the means to monitor and control project activities in real-time.

Facilitate the System Requirements Definition Process

Another motivation for the development of IDEF3 was the need to provide the concepts, syntax, and procedures for building *system requirements descriptions.* These descriptions must be adequately detailed to determine whether a delivered system is acceptable. This requirement implies that the IDEF3 method must support descriptions of the following items.

1. Scenarios of organizational activities.

2. Roles of user types in these organizational activities.

3. User scenarios or user interaction with the information system at the user-function level.

4. System response to user functions.

5. User classes and delineation of user classes.

6. Declaration of timing, sequencing, and resource constraints.

7. User interface objects (e.g., menus, keywords, screens, and displays).

Support Coordinated Activity and Integration of Effort

The process view of business systems is critical to the business decision-making process. This view does need to be coordinated, however, with other views (e.g., the function, information, and organization views). Recognizing this need, the IDEF3 method was designed explicitly to work well both independently and jointly with other methods which address different areas of concentration (e.g., the IDEFØ Function Modeling method) as a complementary addition to the IDEF method family.

Design Features of IDEF3

The needs expressed above levied special requirements on the IDEF3 team of method designers. Given the need to provide an efficient process knowledge capture and display mechanism, IDEF3 was designed to:

1. Be easy to learn and use by individuals having little or no training in structured techniques while promoting consistent, reliable practice.

2. Store, manage, and reuse process information for a variety of downstream uses.

3. Provide an effective means for rapid, reliable, and cost-effective management of both individual and team-based applications.

4. Enable users to readily recognize key differences between alternative process architectures.

5. Enable tailored application to small- and large-scale efforts, and collect process information at both coarse- and fine-grained levels of detail.

6. Function well across varying system domains (e.g., engineering, manufacturing, logistics, and business systems domains).

7. Support integration of effort when applied with other IDEF methods.

IDEF3 achieves these goals by focusing user activity on capturing *descriptions* of how a particular system operates, making it easier to use than traditional modeling methods and enabling maximized downstream reuse of the collected process information. This feature of IDEF3 allows users to concentrate efforts on collecting observations and beliefs about how business systems operate without having to concern themselves with the time-consuming and decision-intensive creation of idealizations characteristic of modeling. In contrast, methods that presume the intent to do modeling are designed to assist in the development of mathematical idealizations, or *models*, that predict what a system will do under a predefined set of conditions.

The distinction between *descriptions* and *models*, though subtle, is an important one. In the context of Integration Definition methods, these terms have a precise technical meaning. The term description is used as a reserved technical term to mean records of empirical observations; that is, descriptions record knowledge that originates in or is based on observations or experience. The term model is used to mean an idealization of an entity or state of affairs. That is, a model constitutes an idealized system of objects, properties, and relations that is designed to imitate, in certain relevant respects, the character of a given real-world system. Frictionless planes, perfectly rigid bodies, the assumption of point mass, and so forth are representative examples of models.

The power of a model comes from its ability to simplify the real-world system it represents and to predict certain facts about that system by virtue of corresponding facts within the model. Thus, a model is a designed system in its own right. Models are idealized systems known to be incorrect but assumed to be close enough to provide reliable predictors for the predefined areas of interest within a domain. The true benefit of models stems from the speed and low cost with which relevant aspects of a real or proposed system can be evaluated. However, the usefulness of a model is limited to the range of questions addressed by its design and the reliability of its approximations in differing contexts.

A description, on the other hand, is a recording of facts or beliefs about something within the realm of an individual's knowledge or experience. Such descriptions are generally incomplete; that is, the person giving a description may omit facts that he or she

believes are irrelevant, or which were forgotten in the course of describing the system. Descriptions may also be inconsistent with respect to how others have observed situations within the domain. IDEF3 accommodates these possibilities by providing specific features enabling the capture and organization of alternative descriptions of the same scenario or process (See Figure 1-1). Modeling necessitates taking additional steps beyond description capture to resolve conflicting or inconsistent views. This, in turn, generally requires modelers to select or create a single viewpoint and introduce artificial modeling approximations to fill in gaps where no direct knowledge or experience is available. Unlike models, descriptions are not constrained by idealized, testable conditions that must be satisfied, short of simple accuracy.

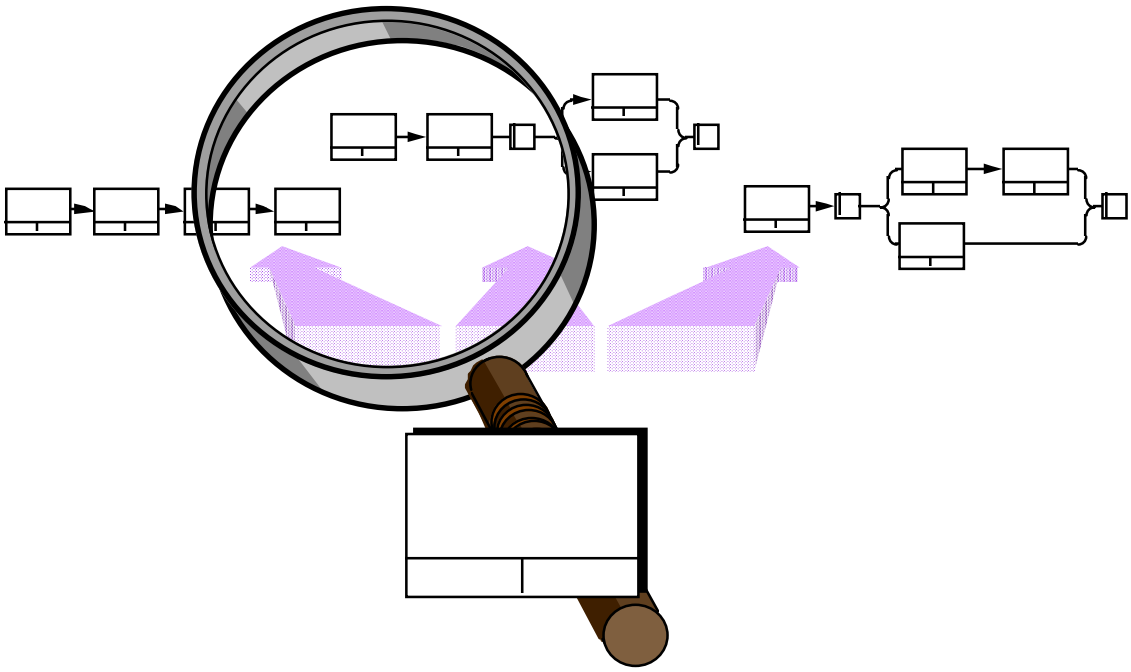

Figure 1-2
IDEF3 Captures Multiple Viewpoints of a Process

The purpose of description capture may be simply to record and communicate process knowledge or to identify inconsistencies in the way people understand how key processes actually operate. By using a description capture method users need not learn and apply conventions forcing them to produce executable models (e.g., conventions ensuring accuracy, internal consistency, logical coherence, non-redundancy, completeness). Forcing users to model requires them to adopt a model design perspective and risk producing models that do not accurately capture their emperical knowledge of the domain.

Description capture may also be undertaken to produce models. Whether accomplished implicitly or explicitly, descriptions are the raw material from which models are made. Thus, the utility of descriptions may also be realized through their reuse in constructing multiple idealizations or models.

Interestingly, models are a form of description. The reverse, however, is not true. A description is not a model. Models are exercised to create analysis data that is not available in descriptions. Unlike models, descriptions do not create analysis data; they may, however, serve as one form of analysis data. For example, descriptions of bus routes and arrival times may be useful forms of data for developing a model of the public transportation system but they do not themselves constitute that model. Similarly, descriptions of an automobile, while potentially useful for other purposes, cannot be used to generate finite element analysis data.

When compared to model building, description capture is attractive as a strategy for knowledge acquisition for several reasons. First, practitioners generally require less training to produce descriptions, rather than models, of their domains. Second, a description of a given situation can easily be reused for a variety of purposes, including model building (e.g., function models, simulation models). IDEF3 is a description organizing and capture method that directly addresses these needs.

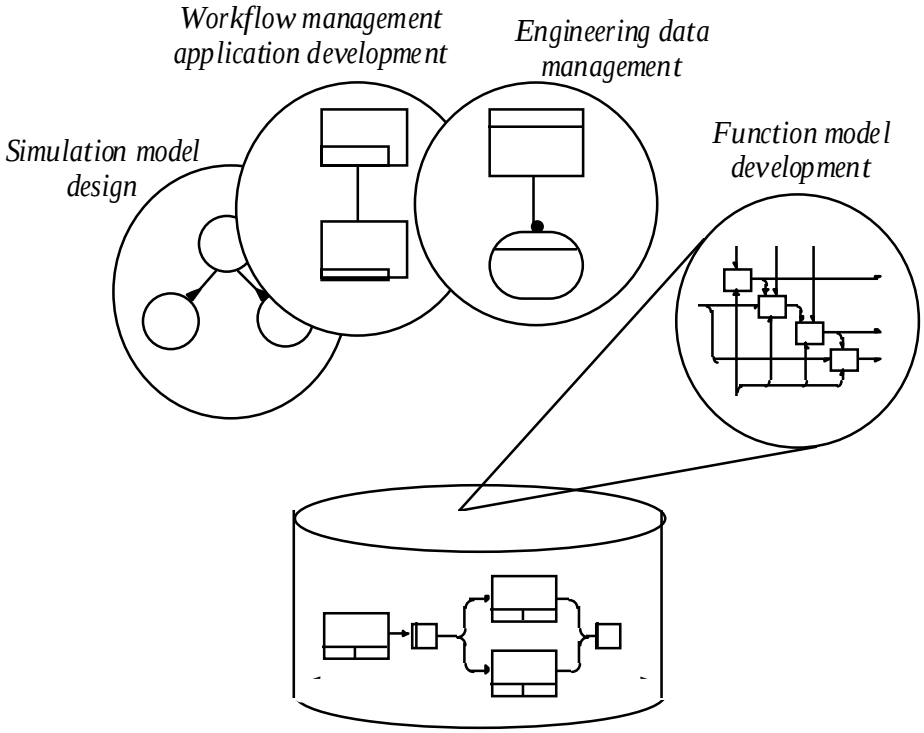

Figure 1-3
IDEF 3's Focus on Description Capture Enables Maximized Reuse

Applicability

IDEF3 has been successfully applied in subject areas spanning all segments of the enterprise. IDEF3 has also been designed to be useful throughout the system development and business evolution process, as illustrated in Figure 1-3.

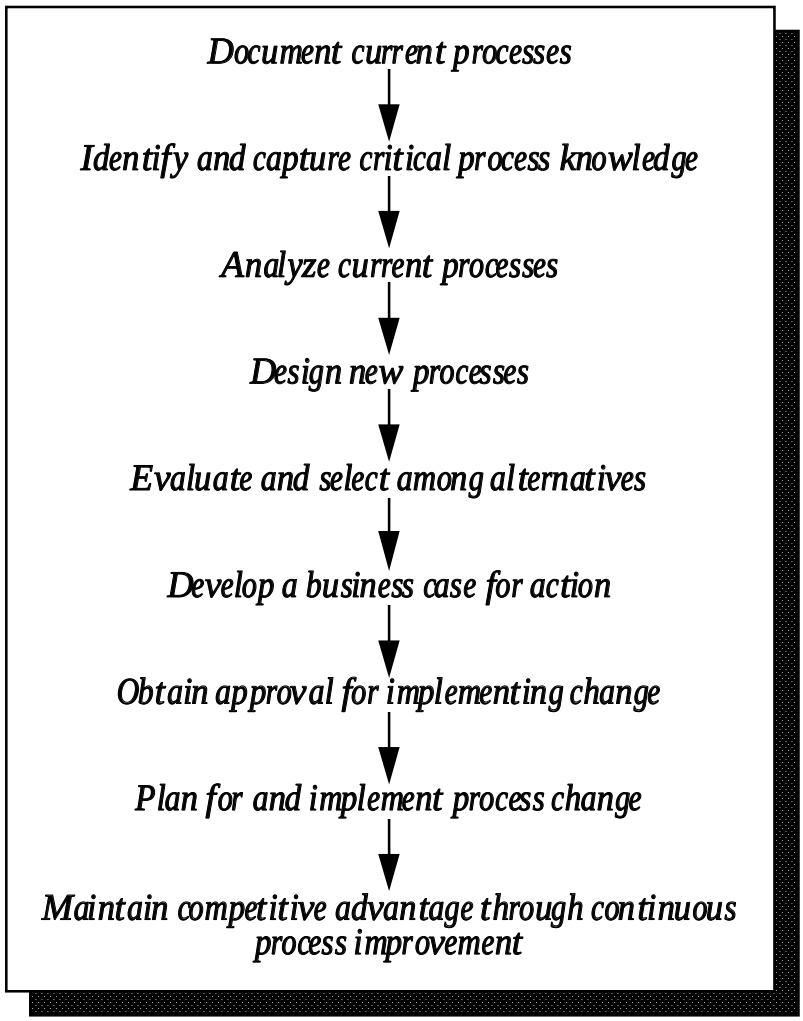

Figure 1-4
IDEF3Can Facilitate Business Improvement

Benefits

Benefits previously realized through the application of IDEF3 can be measured in terms of cost savings, schedule gains, quality improvements, organic capability improvements, and lasting changes to organizational culture. IDEF3 has been used to:

1. Identify obscure process links between organizations.

2. Highlight redundant and/or non-value-added activities.

3. Rapidly design new processes.

Additional benefits gained through IDEF3 have been realized through its usefulness as a mechanism to:

1. Capture and distribute detailed manufacturing process knowledge (e.g., Hubble telescope mirror fabrication process) among geographically dispersed units.

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

3. Provide an implementation-independent specification for human-system interaction.

4. Define data configuration management and change control policy.

5. Document the decision procedures affecting the states and life cycle of critical shared data.

6. Speed the development of high quality IDEF∅ function models.

7. Speed the development and validation of simulation models.

8. Develop real-time control software by providing a mechanism to clearly define facts, decision points, and job classifications.

9. Define the behavior of workflow management systems and applications.

10. Prescribe the process by which change within an organization will be achieved.

Document Organization

This document is divided into the following eight sections:

Section 1	Introduction
Section 2	IDEF3 Overview
Section 3	IDEF3 Process Description Language
Section 4	Developing IDEF3 Descriptions
Section 5	IDEF3 Development: Material Ordering Process Example
Section 6	Understanding IDEF3 Process Descriptions
Appendix A	IDEF3 Elaboration Language
Appendix B	Glossary of Terms

A brief method overview is presented in Section 2, including a description of the basic building blocks used to develop IDEF3 Process Descriptions. Section 3 presents a detailed discussion of the IDEF3 language and its semantics together with advanced concepts for experienced users. Sections 4 and 6 offer practical guidelines for systematically applying the method. A detailed example is described in Section 5. Appendix A describes the IDEF3 elaboration language and Appendix B defines the principal terminology of the IDEF3 method. The elaboration language is computer-processable medium for reusing the information captured through IDEF3 application.

The authors anticipate the use of this document for a wide variety of purposes. Thus, the material is presented in a manner that allows readers to obtain information without having to read the entire document. The following guidelines are suggested.

1. For an *executive overview,* read Sections 1 and 2.

2. To become *proficient* in the development of accurate IDEF3 Process Flow Descriptions, read the entire manual. Place special emphasis on Sections 2, 3, 4, and 6.

3. *Experienced* IDEF3 analysts can use Section 3 as a language reference.

4. To become proficient in *reviewing* IDEF3 Process Descriptions, read Section 6 in detail and browse Sections 2 and 3.

5. An IDEF3 *project leader* should study Sections 3 and 4 in detail, but must also have an understanding of the method in its entirety.

Summary

IDEF3 is designed to assist those engaged in capturing and analyzing the vital processes of an existing or proposed system. Guidelines and simple-to-use graphical language structures aid users in successfully capturing and organizing process information for multiple downstream uses. IDEF3's unique design includes the ability to capture and structure *descriptions* of how a system works from multiple viewpoints, thereby enabling users to capture information conveyed by knowledgeable experts about the behavior of a system rather than directing user activity toward constructing engineering *models* to approximate system behavior. This feature is among the central characteristics distinguishing IDEF3 from alternative offerings. As an integral member of the IDEF family of methods, IDEF3 works well in independent application or in concert with other IDEF methods to identify and develop the vital processes of a business.

IDEF3 OVERVIEW

This section provides a broad overview and examples of the IDEF3 method. Because any discussion of the organizing structures requires references to the basic IDEF3 elements,[3] these will be referred to but not fully defined until Section 3, «IDEF3 Process Description Language.»

Scenarios: The Organizing Structure for IDEF3 Process Descriptions

The notion of a *scenario* or story is used as the basic organizing structure for IDEF3 Process Descriptions. A scenario can be thought of as a recurring situation, a set of situations that describe a typical class of problems addressed by an organization or system, or the setting within which a process occurs. Scenarios establish the focus and boundary conditions of a description. Using scenarios in this way exploits the tendency of humans to describe what they know in terms of an ordered sequence of activities within the context of a given scenario or situation. Scenarios also provide a convenient vehicle to organize collections of process-centered knowledge.

[3] IDEF3 elements are the basic language constructs of IDEF3, including UOBs, junctions, links, object states, referents, and so forth.

Since the primary role of a scenario is to bind the context of an IDEF3 Process Description, it is important to name it appropriately. Scenario names often take the form of an imperative (e.g., verb or verb phrases like *Issue Purchase Order*, *Test Fit*, and so forth) and at times may take the form of a gerund (e.g., a verb that functions as a noun like *Performing Consistency Checks*). A well-chosen scenario name will ensure that the users of the description make the appropriate associations with the real-world situations being described. Correctly identifying, characterizing, and naming scenarios is a necessary step to creating process-centered IDEF3 Process Descriptions. The following examples are typical scenario names.

1. Develop Die Design for Side Aperture Panel

2. Processing a Customer Complaint

3. Implement Engineering Change Request

An IDEF3 Process Description is developed using two knowledge acquisition strategies: a *process-centered* strategy and an *object-centered* strategy. The process-centered strategy organizes process knowledge with a focus on processes and their temporal, causal, and logical relations within a scenario. The second dimension organizes process knowledge with its focus on objects and their state change behavior in a single scenario or across multiple scenarios.

Using one or both of these process knowledge acquisition strategies, IDEF3 users develop IDEF3 Process Descriptions. Both strategies use the basic elements of the IDEF3 language to capture and express the assertions that form the description. Graphical projections of the information contained in process descriptions are created using IDEF3's graphical language. These graphical projections—used to both record process information directly and as a mechanism to display process information—are called *schematics*.

Two types of IDEF3 schematics parallel the two process knowledge acquisition strategies. The IDEF3 *Process Schematic* displays a process-centered view of a scenario. *Object Schematics* support the graphical display of object-centered information. Object Schematics that display an object-centered view of a single scenario are called *Transition Schematics*. Transition Schematics that display additional objects and object relations to provide context-setting information are called *Enhanced Transition Schematics*. Object Schematics that display object-centered information spanning multiple scenarios are simply called *Object Schematics*.

An IDEF3 Process Description may contain zero or more Process Schematics and zero or more Object Schematics. For example, recording that a particular object is recognized by participants in a domain is considered part of the description of that domain. An object so identified may or may not have an Object Schematic associated

with it in a description. Yet these objects are considered part of the description. The scenario concept is used to organize both the process-centered and object-centered views. The collection of scenarios and the information they serve to organize *is* the IDEF3 Process Description.

The following two sections briefly introduce the description representation concepts and syntax available in the two types of IDEF3 schematics.

Process-Centered Views: The Process Schematics

IDEF3 Process Schematics are the primary means for capturing, managing, and displaying process-centered knowledge. These schematics provide a graphical medium that helps domain experts and analysts from different application areas communicate knowledgeabout processes. This includes knowledge about events and activities, the objects that participate in those occurrences, and the constraining relations that govern the behavior of an occurrence.

A process-centered description is constructed systematically, using the basic building blocks of the IDEF3 schematic language, linked together in different ways. These building blocks have specific semantics associated with them. That is, they are used to represent certain kinds of activities or relations in the real-world. A detailed specification of these building blocks is given in Section 3. In this section, some of the important building blocks are introduced, along with an example illustrating how they are used to develop IDEF3 Process Schematics.

The example shown in Figure 2-1 depicts a Process Schematic of the scenario entitled, *Order Material*. In IDEF3, scenarios bound the context of descriptions and are convenient artifacts for describing similar situations from different perspectives. In this example, the owner of a business used IDEF3 to document the material ordering process to assist with new worker training and to enforce company purchasing standards. In particular, the owner wanted to record how Purchase Requests are processed for the benefit of new employees. When asked to describe the process, the business owner related the following.

> «The first thing we do is request material using a Purchase Request form. Then the Purchasing department either identifies our current supplier for the kind of material requested or sets out to identify potential suppliers. If we have no current supplier for the needed item, Purchasing requests bids from potential suppliers and evaluates their bids to determine the best value. Once a supplier is chosen, Purchasing orders the requested material. Those requesting material must first prepare a Purchase Request. The requester must then obtain the Account Manager's approval, or that of

the designated backup, for the purchase. Purchase Requests submitted for Account Manager approval must include the Account Number for the Project that will fund the purchase. Account Managers, or their designated backup, are responsible for, and must approve, all purchases made against their project accounts. After the Account Manager approves the purchase, an authorization signature may be required. To avoid a potential conflict of interest, the requester cannot be the same individual who approves or authorizes the request. Purchase Requests involving Direct projects require an authorization signature whereas Indirect projects do not. Once all the appropriate signatures are in place, the requester submits the signed Purchase Request to Purchasing. Purchasing then orders the requested material. The Purchase Request is thereafter tracked as an issued Purchase Order.»

Scenario Name: Order Material

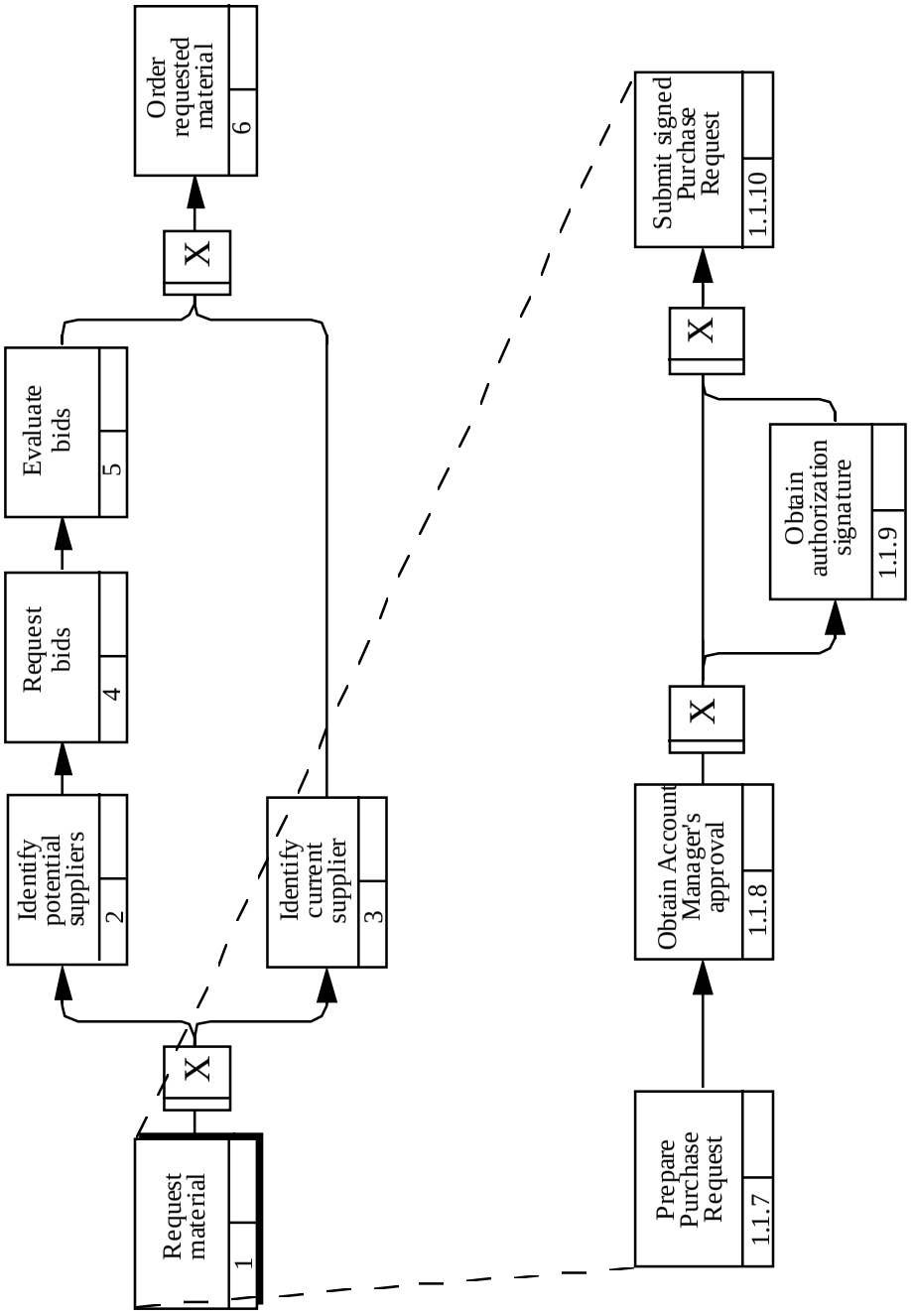

Figure 2-1
Example of a Process Schematic

The processes in the owner's description are represented in the schematic as labeled boxes numbered 1 through 10. Each box represents distinguishable packets of

information about an event, decision, act, or process. That is, boxes represent types of happenings. Such happenings are referred to by the neutral term *units of behavior* (UOBs). Each UOB box represents a real-world process. The information recorded about a UOB includes (1) a name (often verb-based) that indicates what the UOB represents, (2) the names of the objects that participate in the process and their properties, and (3) the relations that hold between the objects. The arrows (called *links*) connecting the boxes in Figure 2-1 indicate the precedence relationships (or more generally *constraints*) that hold between the processes being described. Thus, an instance of the UOB at the source of a link would complete before an instance of the UOB at the end of the same link starts. For example, the UOB labeled *Request bids* would complete before the start of the UOB *Evaluate bids.* The small box containing the «X» denotes a *junction*. A junction is a point in the process where a process splits into multiple paths, or where multiple paths merge. Junctions represent constraints (or the effects of constraints) of the *activation logic* for the process. For example, the first junction in the above figure indicates that only one path will be taken in an activation of the described process.

The IDEF3 method allows users to capture descriptions at varying levels of abstraction by providing a mechanism called a *decomposition.* A decomposition provides a means of organizing a more detailed description of a UOB. The decomposition schematic follows the same syntactic rules as those for a scenario and is created using the same IDEF3 elements. A UOB can have any number of different decompositions, all on the same level. The use of more than one decomposition for the same UOB is for the purpose of representing different points of view or providing greater details of the processing relating to the UOB. The UOB *Request Material* in Figure 2-1 has been *decomposed* into UOBs 7 through 10. The numbers in the lower-left corner of UOB boxes 7 through 10 include a reference to UOB 1 (the first digit) and the decomposition (decomposition 1 of UOB 1). This is illustrative of the IDEF3 numbering scheme which allows explicit traceability between levels of detail in the description. The process description depicted in Figure 2-1 shows the material ordering process from a particular point of view—that of the business owner. It is possible to conceive of other views for this process—for example, that of the Account Manager. Each view to be described would be presented in a separate decomposition with a unique label and number.

The Process Schematic in Figure 2-1 represents a *process-centered view* of the material ordering process. This view focuses on assertions about the processes that occur and their ordering. Sometimes it is convenient to organize the description of a situation from an *object-centered view* (i.e., where a participating object or set of objects is the focus of attention). The next section describes how IDEF3 facilitates process description capture using an object-centered view.

Object-Centered Views: The Object Schematics

IDEF3 Object Schematics capture, manage, and display *object-centered* descriptions of a process—that is, information about how objects of various kinds are transformed into other kinds of things through a process, how objects of a given kind change states through a process, or context-setting information about important relations among objects in a process.

In IDEF3, an object is any physical or conceptual thing that is recognized and referred to by participants in the domain as a part of their descriptions of what happens in their domain. Correctly identifying, characterizing, and naming objects is a necessary step in the creation of object-centered IDEF3 Process Descriptions. Object names are often nouns or noun phrases that may or may not be coupled with a state descriptor. Below are some typical examples.

1. Water: Boiling
2. Purchase Order: Approved
3. Chassis

Object Schematics may be developed in the context of a single scenario, thus characterizing the state transitions traversed by participating objects in an occurrence of the scenario.. These schematics, called Transition Schematics, allow users to specify the rules that govern the transitions between object states in a scenario occurrence. Alternatively, Object Schematics may evolve in a more opportunistic fashion, capturing descriptions of objects, object states, and their transitions across multiple scenarios. Object Schematics developed in this fashion make no attempt to define the structure for object state change behavior in a scenario occurrence. This cross-scenario Object Schematic development approach is often useful when exploring what object-centered process information merits a more detailed focus or when attempting to discover context-setting information about the objects encountered in a description. Object Schematics may be distinguished from the more specialized Transition Schematics (and Enhanced Transition Schematics) by the absence of a context-setting scenario name Generally speaking, IDEF3 Object Schematics are developed to provide an object-centered description of a particular process or scenario. Transition Schematics therefore tend to dominate the attention of those developing IDEF3 Object Schematics.

The schematic in Figure 2-2 represents an Object Schematic for the *Order Material* scenario derived from the business owner's description. This example happens to illustrate a Transition Schematic since it characterizes the nature and structure of object state transitions for occurrences of the Order Material scenario.

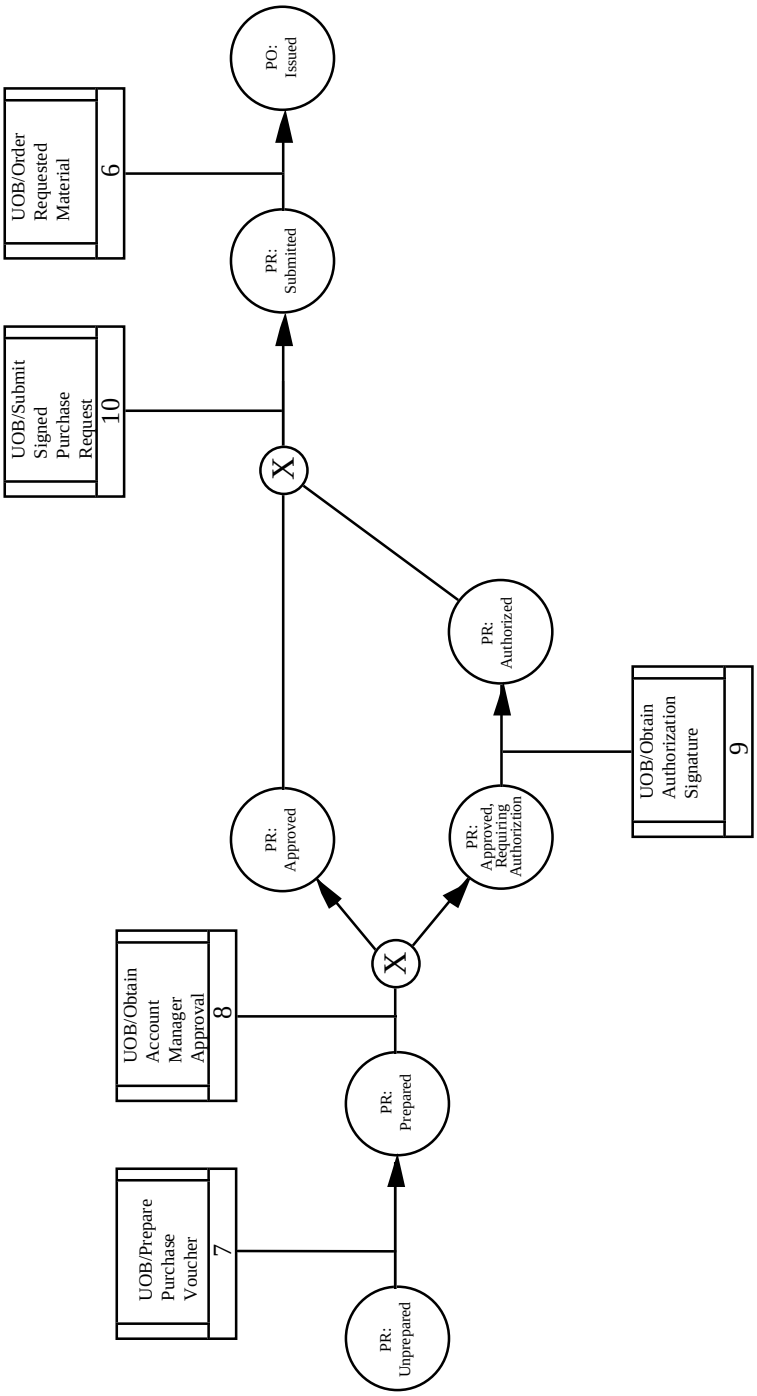

Figure 2-2
Example of a Transition Schematic

A key document in this process is the *Purchase Request* form. This form is eventually transformed into a Purchase Order (PO) via the *Order Material* process. A

circle containing the name of an object represents an object of a certain kind (e.g., Purchase Request, Account Manager, Project). These labeled circles are known as *kind* symbols. A certain kind of object being in a certain state is represented by a circle with a label that captures both the kind itself and a corresponding state, thereby representing the type (or class) of objects that are in that state (within a given process). For example, an approved Purchase Request (PR) would be indicated by the label **PR:approved**, an authorized PR by **PR:authorized**, and so on. One of the first steps to develop an Object Schematic is to identify the possible states in which the object can exist. Though a real-world object often evolves through a continuum of states, an Object Schematic focuses on those distinguished states of particular interest to the domain expert. The transition arcs (arrows with triangular, filled-in heads) connecting the circles symbolize a *state transition*, the activity of changing from one state to another. The conditions that establish when an object is in a given state, how it exists a state, how it can transition between states, and how it can enter a new state are recorded on a special form. The banded boxes linked to the arrows (called *referents*) are aids to describe the relationships between objects states and UOBs, scenarios, or other Transition Schematics that participate in a scenario occurrence. For example, during the transition of the object *PR* from its state of having been prepared for review by an Account Manager (i.e., **PR:prepared**) to an approved state (i.e., **PR:approved** or **PR:approved requiring authorization**), the process represented by the UOB *Obtain Account Manager approval* must initiate and complete. The transition junctions containing an «X» (for *exclusive or*) indicate the choice of exactly one path among several possible paths in an occurrence.

Thus, Figure 2-2 indicates that Purchase Requests transition from an *unprepared* to a *prepared* state and from a *prepared* state to either an *approved* state or an *approved requiring authorization* state. If the Purchase Request requires authorization, it will transition to an *authorized* state before transitioning to a *submitted* state. Otherwise, it will transition directly to the *submitted* state. After the Purchase Request reaches the *submitted* state, the object will transition to an *issued* Purchase Order. UOBs, scenarios, and other Transition Schematics that participate in a transition between states are indicated by attaching appropriately labeled referents to the Object Schematic. The relative positioning of referents on the Transition Schematic indicates the order in which they *occur*. For example, the position of the UOB *Prepare Purchase Request* in Figure 2-2 indicates that it initiates and completes before all other UOBs referenced by the schematic in an occurrence of the scenario.

It is interesting to note that among the possible state transitions represented, none reflect a failed request. This is simply because the original dialog contained no information about such situations. This is a key point in the use of IDEF3. IDEF3 is intended as a mechanism for structuring the assertions made by the domain expert. It does not force the completion of partial information with *modeling* assumptions.

The schematic in Figure 2-2 may be embellished to include additional context-setting information. An example of this is provided in Figure 2-3. In this figure, the Transition Schematic has been supplemented with objects and appropriate relation links that provide additional information. For example, the three-place relation that stands between the kinds **PR: Prepared**, **Direct Project**, and **Authorization Signature** indicates that Purchase Requests involving Direct Projects require an authorization signature. Furthermore, an Authorization Signature is included on each Purchase Request that has been authorized. The schematic also indicates that a Requester may not be the same individual who approves or authorizes a Purchase Request.

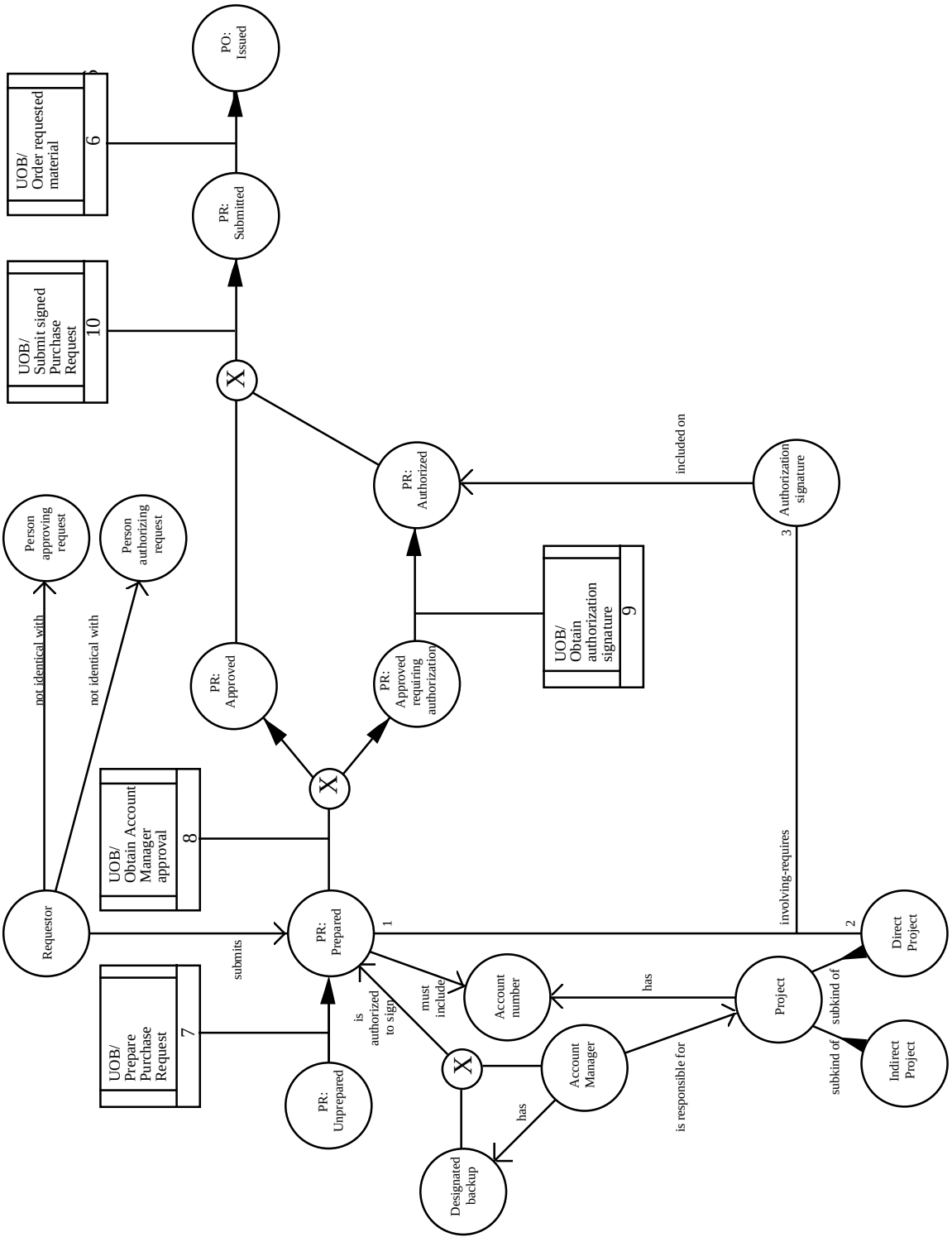

Figure 2-3
Example of an Enhanced Transition Schematic

Section 5 contains a more detailed exposition of the example provided here. In particular, the step-by-step process used to develop both the Process and Object Schematics is provided. For a more detailed description of the basic IDEF3 schematic elements and their semantics, readers are invited to continue with Section 3.

IDEF3 PROCESS DESCRIPTION LANGUAGE

The following sections describe the basic elements of the IDEF3 process description language and how those elements can be combined todynamic form semantically rich descriptions of dynamic systems. An IDEF3 process description organizes the network of relations between situations in a specified scenario. IDEF3 descriptions are developed from two different perspectives: process-centered and object-centered. Because these approaches are not mutually exclusive, IDEF3 allows cross-referencing between them to represent complex process descriptions. Sections 3.1 through 3.5 contain descriptions of the syntactic elements of the IDEF3 process description language. Section 3.6 describes how to combine those elements to form process descriptions. Section 3.7 contains descriptions of the syntactic elements of the IDEF3 object state transition description language, which enables one to construct object-centered descriptions of processes. The mechanisms for cross-referencing among statements made in each of these languages are introduced as part of the individual language specification. Examples are interspersed throughout these sections to illustrate how the basic syntactic elements are combined to build IDEF3 schematics.

Basic Elements of IDEF3 Process Descriptions

The basic syntactic elements of the IDEF3 process description language are shown in Figure 3-1a. Figure 3-1b displays alternative symbol conventions for first-order relations.

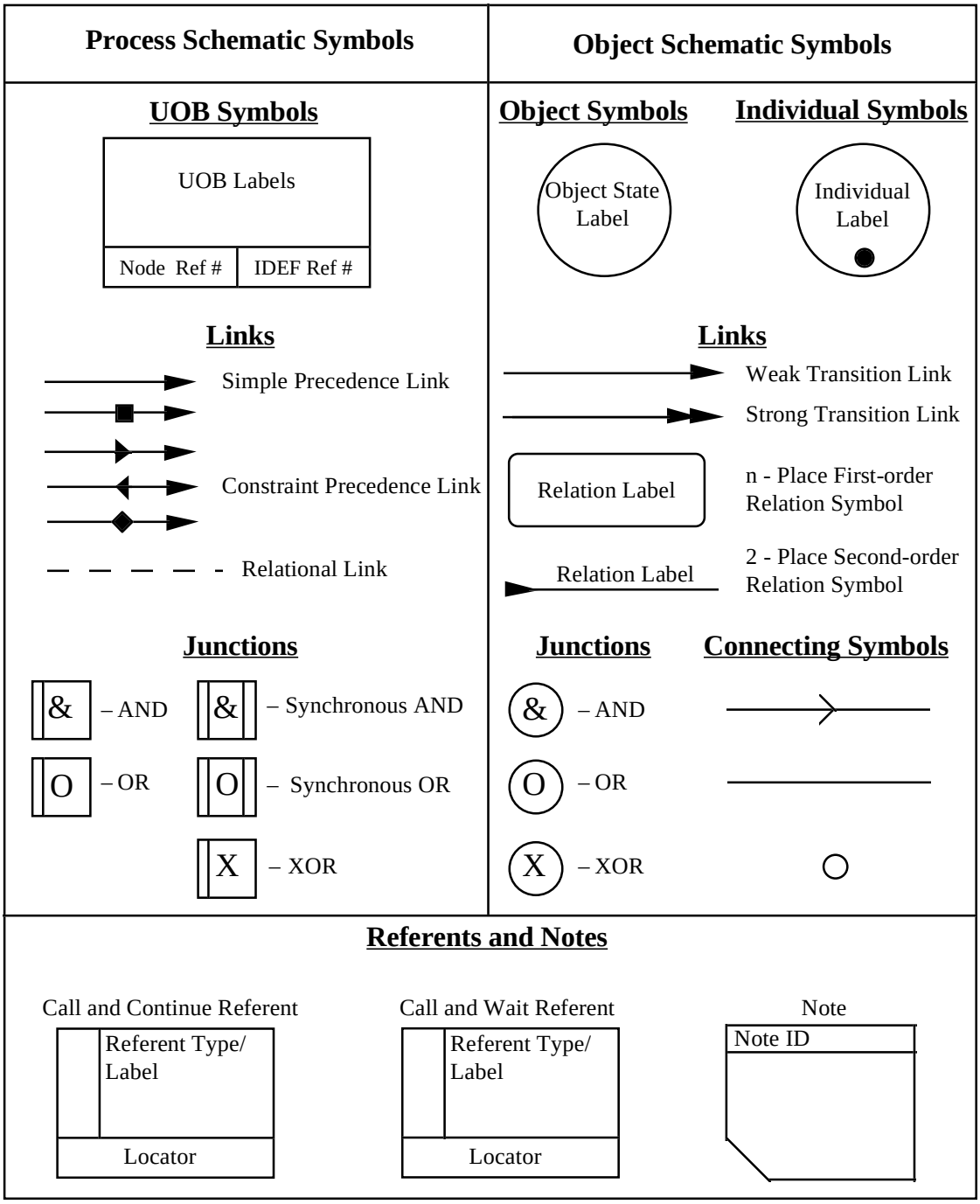

Figure 3-1a
Symbols Used for IDEF3 Process Description Schematics

Figure 3-1b
Alternative Symbol Conventions for First-Order Links

The informal syntax and semantics of these symbols and the more complex structures that can be constructed from them are presented in the following sections.

Process Schematics

Process schematics tend to be the most familiar and broadly used component of the IDEF3 method. These schematics provide a visualization mechanism for process-centered descriptions of a scenario. The graphical elements that comprise process schematics include Unit of Behavior (UOB) boxes, precedence links, junctions, referents, and notes. Referents and notes are constructs that are common across process and object schematics. Each of the graphical elements used for developing process schematics is presented below, together with discussions of how to formulate more complex statements using those graphical elements. The discussion begins with the most fundamental of these building blocks: the UOB.

Units of Behavior

To be clear about the meaning of UOB (and, hence, the meaning of a UOB *box*) a distinction must be made between *types* and *instances*. The distinction is familiar in the field of database design. To design a database schema, a database designer abstracts away from the particular objects found in a given system and isolates the *classes*, or *types*, of which those objects are *instances*. Similarly, the designer abstracts away from the particular attributes of those objects to identify attribute types (e.g., color, model, hardness) common to all instances of the same type. This information is then used to design the relation schema for a particular type of object about which one wishes to keep information. This is the kind of information expressed by a database schema in the Entity Relationship (ER) or IDEF1X modeling language; it describes the *types* of objects in a

given system, the *types* of attributes objects of any given type exhibit, and the logical constraints that bind them together.

By the same token, when one captures «what's going on» in a given system, one describes *not* what in fact happened in the system at particular time, but rather what happens *in general* in the system; one abstracts the general dynamic *patterns*, the general *types* of situation, that can occur again and again in the system. A *UOB*, then—e.g., a *Planning Activity*, or *Make or Buy Decision*, or *Contract Award Event*—describes a *type* of situation; an *instance* of a UOB is simply an occurrence of the UOB. Like a database schema, a process description describes a system at the type level. A process description represents the types of situations (processes, functions, etc.) that can occur in the system and the logical and temporal constraints that bind them together.

As illustrated in Figure 3-1a, a UOB is represented by a special kind of box with a unique *label*. Though it is important to bear in mind the distinction between a UOB box and the UOB it stands for (just as it is important to distinguish a name from the person the name stands for), in practice, the term «UOB» is often used ambiguously to refer, at some times, to a given UOB box within a schematic, and at other times, a given UOB in the scenario represented. Context is usually sufficient to determine which is meant on a given occasion. Many times, because of the structural similarity between a schematic and the scenario it represents, the ambiguity doesn't matter.

Links

Links are the glue that connect UOB boxes to form representations of dynamic processes. Links are used primarily to denote significant relationships among UOBs. Links draw attention to important relations between UOBs in a process. Examples of the types of relations that can be highlighted by IDEF3 links include temporal, logical, causal, natural, and conventional. However, the vast majority of the time, users are most interested in indicating simple temporal precedence. Hence, a special class of links is devoted to expressing this relation. The *precedence link elaboration document*, enables users to capture additional details about a particular precedence link. Links are drawn to start or terminate at any point on a UOB box or junction symbol. There are two basic types of links used in IDEF3 process schematics: *precedence* links and *dashed* links. The symbols that represent each type are shown in Figure 3-2.

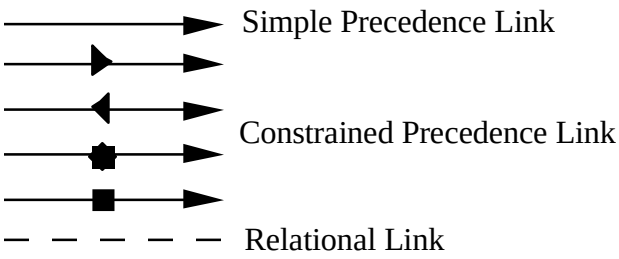

Figure 3-2
IDEF3 Link Types

Simple Precedence Links

Precedence links express temporal precedence relations between instances of one UOB and those of another. They are the most widely used link and are denoted by a solid arrow, perhaps with an additional marker attached to the stem of the arrow, as indicated in Figure 3-2. Precedence links connect UOB boxes, as illustrated in Figure 3-3, with a simple precedence link.

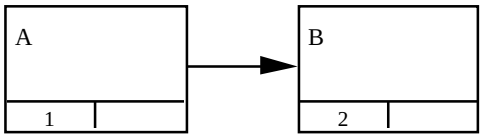

Figure 3-3
Basic Precedence Link Syntax

Box 1, (labeled «A») at the «back» end of the link is known as the *source* of the link and box 2 (labeled «B») at the «front» end of the link is known as the *destination*. Considered as an IDEF3 schematic, box 1 is known as the (immediate) *predecessor* of box 2 in the schematic, and box 2 the (immediate) *successor* of box 1.

The meaning of this schematic, as with IDEF3 schematics generally, can be understood in terms of possible *activations* of the schematic. An activation of a schematic is a collection of instances of some or all of the UOBs in the process represented by the schematic whose temporal and logical properties satisfy the conditions specified in the schematic. In general, there are many different patterns of activation for a given schematic. For example, one possible activation pattern for simple two box schematics like Figure 3-3 is when a single instance *a* of UOB A is followed by an instance *b* of UOB B. More precisely, any pair of instances *a* and *b* of A and B, respectively, where *b* does not start before *a* completes would be a legitimate activation of Figure 3-3.

Activation Plots

Activation plots are used to represent activations. The UOB instances in an activation must occur within a single, finite, interval of time that begins when the first instance in the activation begins, and ends when the last instance ends. To determine whether a given collection of UOB instances is an activation of a given description, it is useful to plot the general pattern of their occurrence over such an interval. For description development purposes, it is often useful to plot the activation pattern for a collection of observations over a given interval in order to discover the general pattern. This can be done by vertically listing the names of the UOBs and plotting the instances of each UOB according to the time and duration of its occurrence. For example, an activation plot of the schematic in Figure 3-3 is shown in Figure 3-4 where the line to the right of each UOB name represents the time interval in which an instance of that UOB occurs. That there is no horizontal overlap in the projections of the two lines indicates that the instance of B does not begin before the instance of A ends, as required by the semantics of Figure 3-3. Hence, the plot does indeed represent an activation of Figure 3-3.

Figure 3-4
Activation Plot for Figure 3-3

Constrained Precedence Links

Figure 3-3, with a simple precedence link, says nothing about whether instances of either UOB can occur in the system being described without a corresponding instance of the other. For all Figure 3-3 says, an instance of A could occur without an instance of B; or an instance of B could occur before an instance of A. The semantics of the simple precedence link is thus rather permissive. Constrained precedence links add constraints over and above the activation semantics of simple precedence. The first of the constrained precedence links indicates that any instance of the source UOB *must* be followed by an instance of the destination UOB. This is what is meant by the «directionality» of the link; the constraint is in force only from «left to right.» So, for example, as with simple precedence, an activation of the schematic in Figure 3-5a consists of an instance of *Sign timesheet* followed by an instance of *Obtain timesheet approval*. However, the constrained link in the schematic also expresses that any instance of *Sign timesheet* must be followed by an instance of *Obtain timesheet approval*. Lack of such an instance indicates an *inconsistency* with the system as described. That is, such a collection of events would not be classified as an activation of the IDEF3 description.

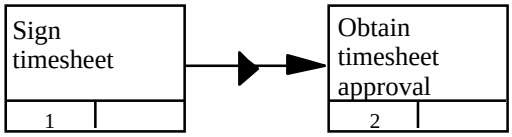

Figure 3-5a
Example of a Schematic Involving a Constrained Precedence Link

Given the directionality of the link in Figure 3-5a, instances of *Obtain timesheet approval* alone are not prohibited by Figure 3-5a; such cases might occur, e.g., when an employee quits before timesheets for a given pay period are turned in (in which case the subsequently approved timesheet was never signed).

Two remaining constrained precedence links are illustrated in Figure 3-5b. These links indicate similar constraints extending in the opposite direction and in both directions, respectively. That is, the top schematic indicates (again, in addition to the activation semantics of simple precedence) that an instance of B must be preceded by an instance of A, and the bottom schematic indicates both that any instance of A must be followed by an instance of B, and that an instance of B must be preceded by an instance of A. These constraints add a *normative* component to the description of a system, i.e., a component that expresses not just how the system has been observed to behave, but how it *ought* to behave. Constrained links are thus particularly useful when IDEF3 is used to *model* a system, not just record beliefs and observations about its behavior.

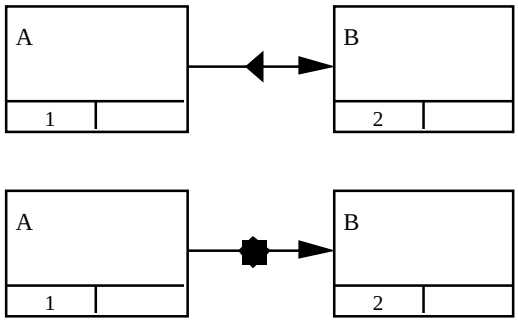

Figure 3-5b
Further Examples of Constrained Precedence Links

Clearly, these three links do not exhaust the possible constraints that might hold between UOBs. For instance, one might wish to add to the simple precedence semantics of Figure 3-3 that no more than five minutes can separate the completion of any instance of A in any activation and the beginning of an instance of B that follows. The final constrained precedence link indicates the presence of general constraints of this sort. For this reason, it is called a *general* constrained precedence link, and is illustrated in Figure 3-6. Because the nature of these constraints can vary widely, they must be recorded explicitly in the precedence link elaboration document. (See Precedence Link Elaboration Document subsection that follows later).

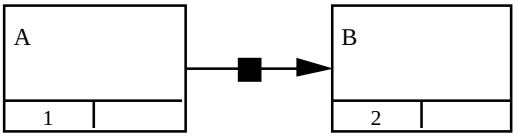

Figure 3-6
General Constrained Precedence Link

Dashed Links

Dashed links carry no predefined semantics. For this reason, they are often referred to as user-defined links or relational links. This type of link highlights the existence of a (possibly constraining) relationship between two UOBs. For example, the relational link in Figure 3-7 might indicate the constraint between *Sign timesheet* and *Obtain timesheet approval* that one cannot approve one's own timesheet. The precise character of the relationship indicated by a relational link is specified in the Relational Link Elaboration document.

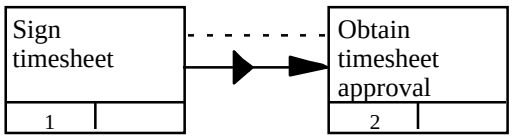

Figure 3-7
Example of a Relational Link

Link Numbers

All links have an elaboration and unique link numbers. Precedence link numbers are prefaced by the letters *PL* (for «precedence link»). Relational links are prefaced by the letters *DL* (for «dashed link»). For example, precedence links may be numbered PL1, PL2, and so on. The uniqueness of link numbers is ensured by using a procedure similar to the UOB numbering scheme. That is, link numbers are assigned sequentially from a pool allocated to an author. Displaying link numbers on the process schematics is optional.

Activation Semantics for Nonbranching Process Schematics

Before introducing junctions (which give IDEF3 the capacity to describe the structure of branching processes), it is useful to generalize the semantics for the different link types for larger, nonbranching schematics. Consider the simple schematic in Figure 3-8 that describes the process of holding a meeting to discuss committee reports.

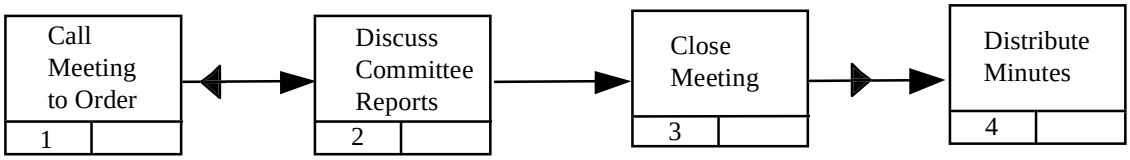

Figure 3-8
Nonbranching IDEF3 Schematic

As with IDEF3 schematics generally, the basic semantics of this schematic is to be understood in terms of the pattern of possible activations it describes. In other words, the schematic specifies exactly what *counts* as a meeting in the given context. As in the simple two box case, an activation will generally exhibit the following pattern: An instance of *Call Meeting to Order* is followed by an instance of *Discuss Committee Reports*, which in turn is followed by instances of *Close Meeting* and *Distribute Minutes*, where each instance in the series begins no earlier than its predecessor ends. As with all nonbranching schematics (and indeed, all schematics without *disjunctive* branches), the typical activation pattern for Figure 3-8 is illustrated in the activation plot in Figure 3-9.

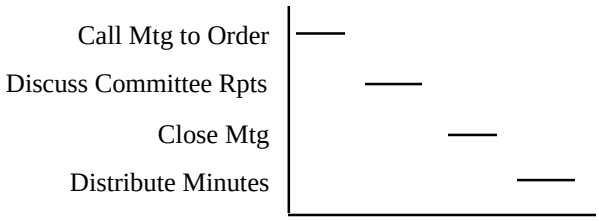

Figure 3-9
Activation Plot for Figure 3-8

The constrained precedence links indicate further constraints on the process: committee reports must not be discussed before the meeting is called to order; and after the meeting, minutes must be distributed. The absence of any constraint between the second and third UOBs, for example, allows for the possibility of a meeting ending before the *Discuss Committee Reports* UOB completes. In such a case, the truncated meeting would be an activation of the described process; it would not violate any constraints, and hence would be consistent with the description. The constraints indicated by constrained links are to be understood as being *independent* of any activation. So even if the meeting is closed without the UOB *Discuss Committee Reports* being completed, the constraint between the last two UOBs nonetheless requires the distribution of minutes after the close of the truncated meeting.

Junctions

Junctions in IDEF3 provide a mechanism to specify the logic of process branching. Additionally, junctions simplify the capture of timing and sequencing relationships between multiple process paths.

Junction Types

IDEF3 schematics are, in general, type-level descriptions of complex processes (i.e., process *types*). Such processes are rarely linear. More typically, they involve any or all of four general sorts of *branch points*:

1. Points at which a process diverges into multiple *parallel* subprocesses;

2. Points at which a process diverges into multiple (possibly nonexclusive) *alternative* subprocesses;

3. Points at which multiple *parallel* subprocesses converge into a single «thread;» and

4. Points at which multiple *alternative* subprocesses in the process converge into a single thread.

IDEF3 introduces four general types of *junctions* to express the four general sorts of branch points. The first two sorts are expressed by *fan-out* junctions: *Conjunctive* fan-out junctions represent points of divergence involving multiple parallel subproceses, while *disjunctive* fan-out junctions represent points of divergence involving multiple alternative subprocesses. The last two sorts of branch points are expressed by *fan-in* junctions: *conjunctive* fan-in junctions represent points of convergence involving multiple parallel subproceses, while *disjunctive* fan-in junctions represent points of convergence involving multiple alternative subprocesses. There is one type of conjunctive junction, or AND junction, indicated by «&». There are two types of disjunctive junctions: inclusive and exclusive junctions, or OR and XOR junctions, respectively, depending on whether the alternatives in question are mutually exclusive. This classification of junctions is depicted in Figure 3-10. Their semantics is discussed more fully in the following sections.

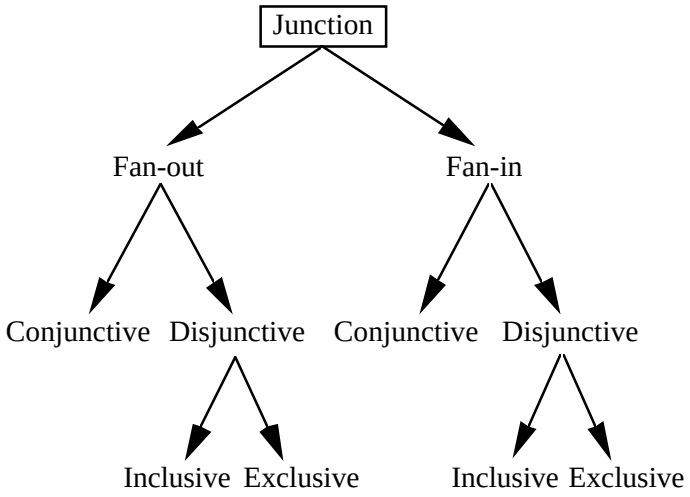

Figure 3-10
Classification of Junction Types

Basic Junction Syntax

Junctions represent branch points in a general process, points at which either a single «thread» in the process diverges into multiple (parallel or alternative) threads, or multiple threads converge into one. In IDEF3, such divergence is represented by a single junction serving as the source of multiple precedence links and convergence by a single junction serving as the destination of multiple precedence links. Divergence to, and convergence from, multiple *parallel* subprocesses are indicated by the use of an AND junction, as illustrated in Figure 3-11.

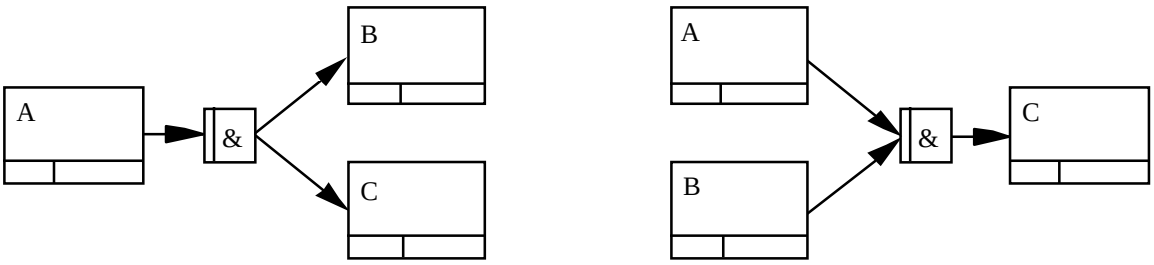

Figure 3-11
Diverging and Converging Parallel Subprocesses

Similarly, divergence to, and convergence from, multiple *alternative* subprocesses are indicated as illustrated in Figure 3-11 except by the use of either an OR or an XOR junction, depending, once again, on whether the alternatives are mutually exclusive.

As a convention, the precedence link coming into a fan-out junction (if there is one) will be drawn without an arrow tip, and the outgoing precedence links in a fan-out

junction will be drawn with a single stem, and with rounded rather than sharp corners. Parallel conventions hold for fan-in junctions. To illustrate, these conventions are applied to the top two schematics in Figure 3-12, yielding the bottom two schematics.

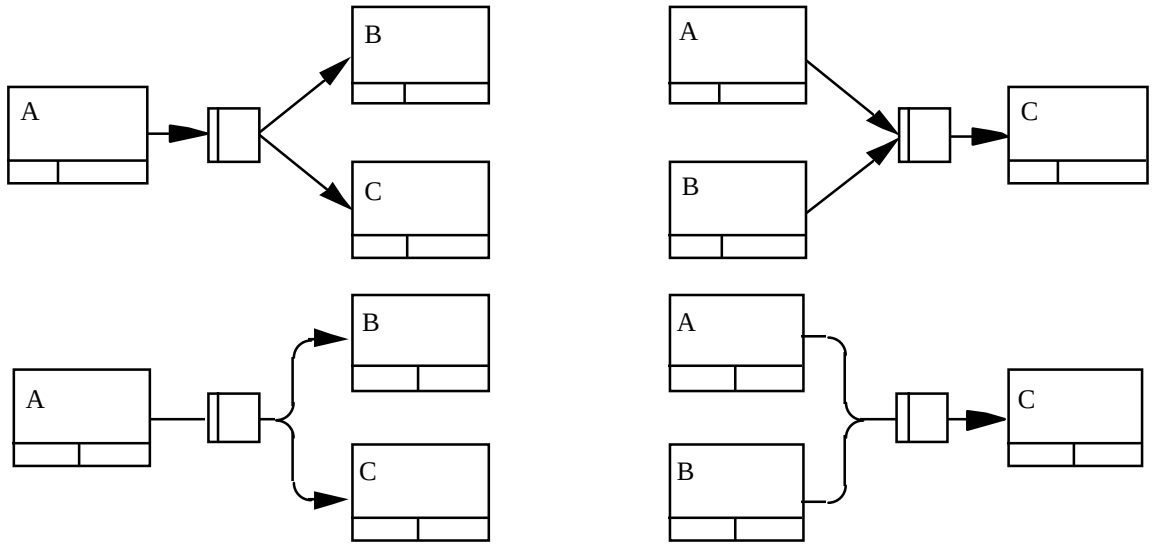

Figure 3-12
Graphical Conventions for Precedence Links Connecting to Junctions

Note that junction symbols are not intrinsically fan-out or fan-in. Rather, a given *occurrence* of a junction symbol in a schematic is fan-out or fan-in depending on whether it is source or a destination, respectively, of multiple paths.

Junction Numbering Scheme

To make unambiguous references to the junctions in an IDEF3 schematic, an identification scheme for IDEF3 junctions is provided. Recall that precedence links are assigned unique numbers beginning with the letters PL. Junction numbers follow an identical numbering scheme, except that junction reference numbers start with the letter J, thus: J1, J2, ..., J*n*. As with links, no two distinct junctions can be assigned the same junction number.

Basic Junction Semantics

A fan-out AND junction in a schematic means that, in an activation of the schematic that reaches the point in the structure of the process represented by that junction, there will be instances of all UOBs denoted by the UOB boxes that are (immediate) successors of the junction. If a *synchronous* AND junction is used, then, to be an activation of the schematic, those instances must all start *simultaneously*. Similarly, the intuitive meaning of a fan-in AND junction in a schematic is that, in an activation of the schematic that

traverses that junction, there will be instances of all UOBs denoted by the UOB boxes that are (immediate) predecessors of the junction. And if a *synchronous* AND junction is used, then, to be an activation of the schematic, those instances must all end simultaneously. Thus, an activation of the left schematic in Figure 3-13 will consist of an instance of UOB A followed by instances of both B and C. Similarly, an activation of the right schematic in Figure 3-13 will consist of an instance of UOB C preceded by instances of both A and B; if a *synchronous* AND junction is used, then, to count as an activation of the schematic, A and B must end simultaneously.

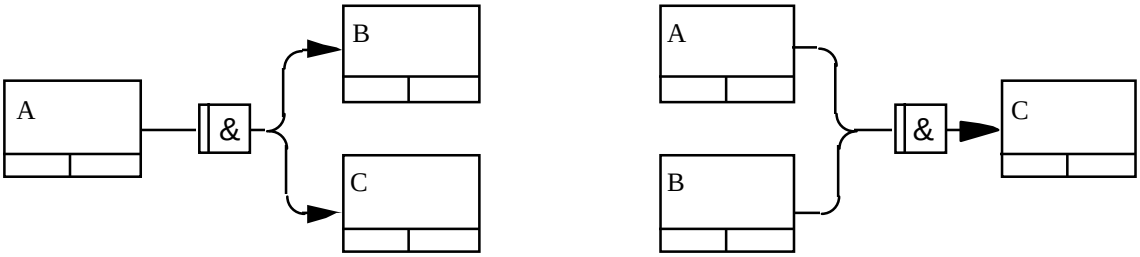

Figure 3-13
Sample Schematics to Illustrate Semantics of AND Junctions

A fan-out OR junction in a schematic indicates that, in an activation of the schematic, there will be an instance of at least one of the UOBs connected to the junction to the right. Similarly, a fan-out XOR junction in a schematic indicates that, in an activation of the schematic, there will be an instance of exactly one of the UOBs connected to the junction to the right. If a *synchronous* OR junction is used, then those instances must all must start simultaneously. (This constraint does not apply to XOR junctions, since there can be only one such instance in an XOR activation.) Likewise, the intuitive meaning of a fan-in OR junction in a schematic is that, there will be at least one instance of the UOBs connected to the junction to the left. If a *synchronous* OR junction is used, then, those instances (if there is more than one) must all end simultaneously. Hence, an activation of the schematic to the left in Figure 3-14 consists of an instance of UOB A followed by an instance of either B or C, or both B and C. Similarly, an activation of the schematic to the right in Figure 3-14 consists of an instance of UOB C preceded by an instance of either B or C, or both. If the schematics in Figure 3-14 used XOR junctions, then legal activations would not include those in which both B and C occur in the first case and both A and B in the second.

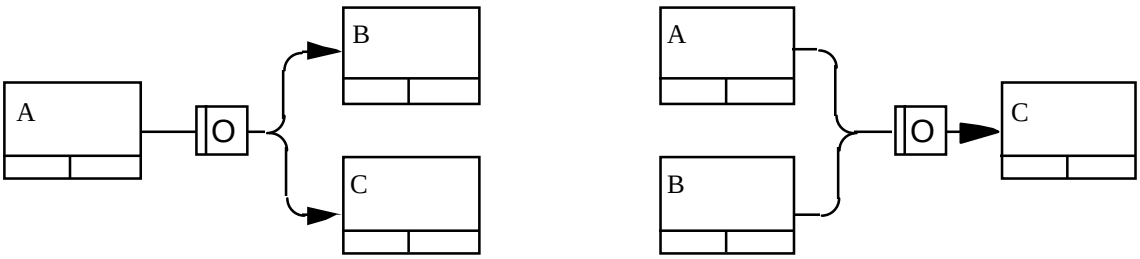

Figure 3-14
Sample Schematics to Illustrate Semantics of OR Junctions

Although not a part of their actual semantics, junctions in an IDEF3 schematic often have an associated «decision logic.» The decision logic of a junction determines the timing and sequencing of the succeeding UOBs. For OR and XOR junctions, the decision logic documents how the process will branch in a given activation. Similar logic is captured for AND junctions (e.g., when the logic involves more than mere synchronicity). The decision logic of a junction is recorded in the elaboration for the junction.

Because of the possibility of both conjunctive and disjunctive branching in a process, branching is never indicated in IDEF3 by the presence of multiple outgoing precedence links from a UOB box. Such a construct is semantically ambiguous between a splitting of the process into concurrent subprocesses or a conditional branch in which only one (or perhaps more) of the branches is instantiated in any given activation. Use of a junction, however, makes the meaning of the branch entirely clear. A similar ambiguity can arise if a UOB box is the destination of multiple arrows, there are cases—often called «loopbacks»—in which this is acceptable.

Junctions are *always* used in IDEF3 to indicate branching in a process; branching is never indicated by linking a single source UOB with multiple destinations by means of several precedence links; such schematics are semantically ambiguous between the three different types of branching that are identified and — purposely! – distinguished in IDEF3.

Combining Junctions

The real power of IDEF3 lies in its ability represent processes in which multiple parallel and alternative threads are woven together into a single complex whole. The key to such complex representations lies in the proper use of junctions, in particular, finding the right combinations of junctions to represent the process in question. Some of the most basic combinations are illustrated in this section.

It is common to find processes in which a single thread diverges into multiple threads and then, at some later point converges back into a single thread. In IDEF3, such

processes are represented by combining fan-out junctions and fan-in junctions. Figure 3-15 represents a process in which a thread diverges into parallel subprocesses and then converges. Because the processes run in parallel, they are represented by AND junctions.

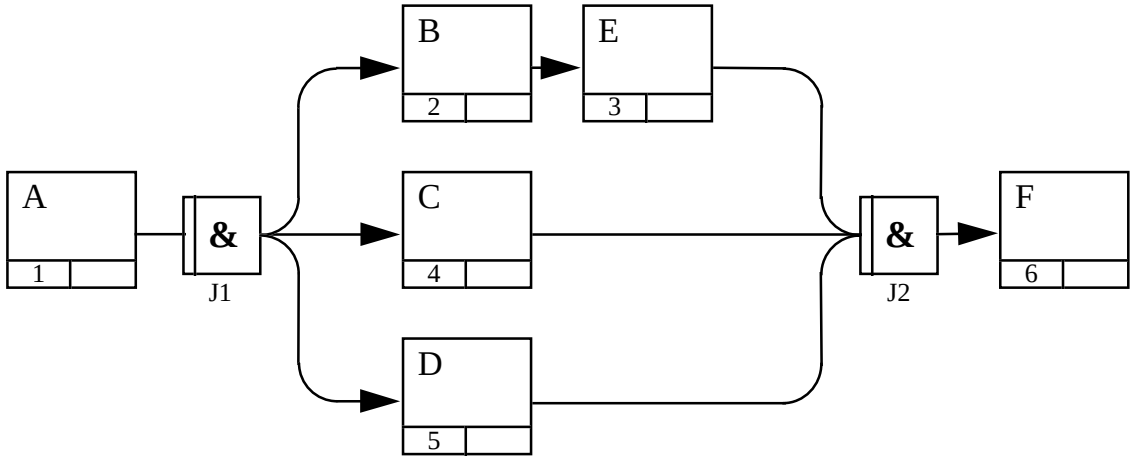

Figure 3-15
Schematic with Asynchronous AND Junctions

Because junction J1 separates UOB box 1 and boxes 2, 4, and 5, in any activation of Figure 3-15, an instance of UOB A will complete before any of the succeeding UOBs are instantiated. An activation of the schematic in Figure 3-15 will proceed in the following manner. After an instance of UOB A, the three UOBs (B, C, and D) will be instantiated. Because J1 is asynchronous, these instances can begin in any order. Because all three paths converge to J2, UOB F will be realized only after the instances of UOBs E, C, and D complete. Because J2 is also asynchronous, no particular order or timing of the completions is implied. This pattern of activation is illustrated by the plot in Figure 3-16.

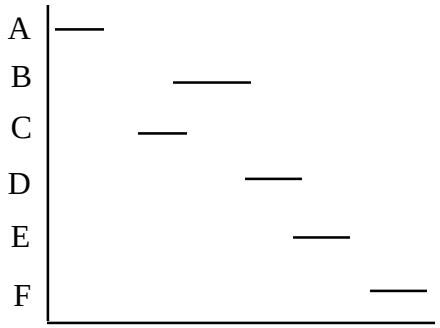

Figure 3-16
Activation Plot for Figure 3-15

As in Figure 3-15, the precedence link L1 shown in Figure 3-17 requires that an instance of UOB A be completed before the UOBs signified by the succeeding boxes can be instantiated. Synchronous logic is indicated by junction boxes having two vertical

bands (Compare Figure 3-15 and 3-17). The synchronous AND junction J1 indicates that, in an activation, the instances of UOBs B, C, and D will initiate simultaneously. Likewise, the synchronous AND junction J2 indicates simultaneous completion of those instances of UOBs B and C and an instance of UOB E before the process continues past the junction to an instance of UOB F.

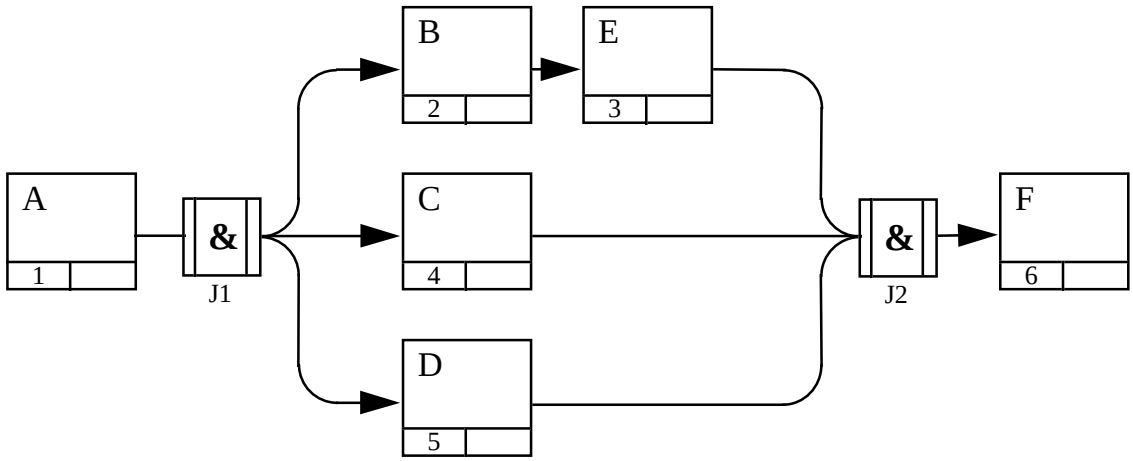

Figure 3-17
Synchronous AND Junctions

Figure 3-18 illustrates the added structure on activations imposed by the synchronicity constraints.

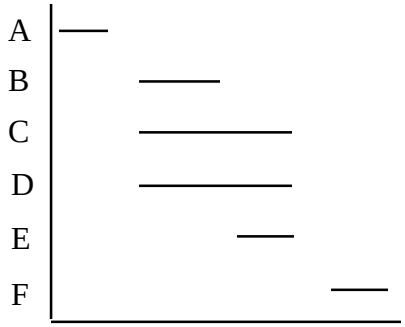

Figure 3-18
Activation Plot for Figure 3-17

Figure 3-19 is structured like Figure 3-15 except that junctions J1 and J2 are asynchronous OR junctions. In an activation of the represented process, J1 indicates that, following an instance of A, one or more of the UOBs B, C, and D will be realized. This will initiate one to three «threads» in the activation. Because J2 is an asynchronous OR junction, only *one* of the threads needs to complete before an instance of F initiates.

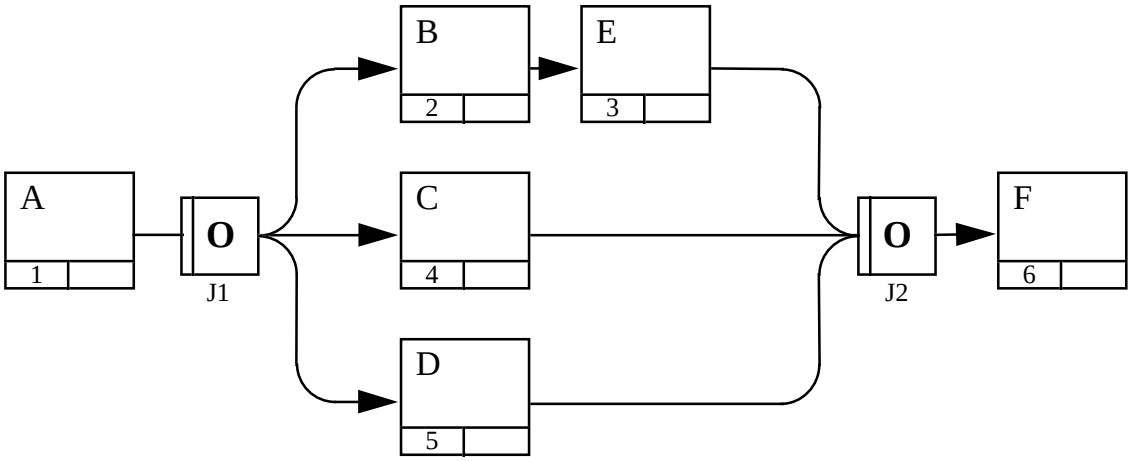

Figure 3-19
Asynchronous OR Junctions

Figure 3-20 illustrates the use of two synchronous OR junctions in combination. The fan-out OR junction implies that, in an activation, instances of one or more of the UOBs B, C, and D will start after an instance of A.

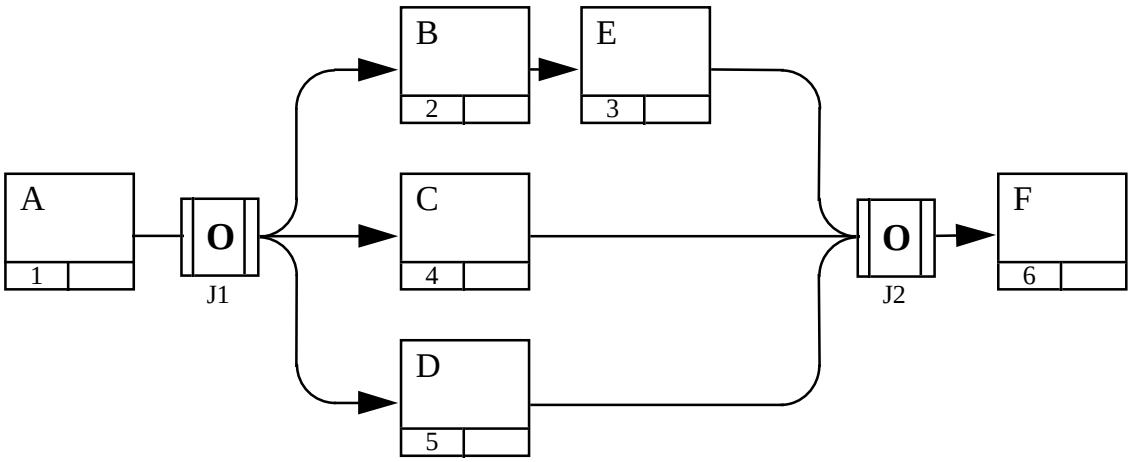

Figure 3-20
Synchronous OR Junctions

Because the junction is synchronous, when more than one UOB is instantiated, the instances occur simultaneously. If one of these is an instance of UOB B, it will be followed by an instance of UOB E, which will compete simultaneously with whatever instances initiated along with the instance of UOB B, as illustrated by the left activation plot in Figure 3-21. An activation in which UOB B is not instantiated is also illustrated by the right activation plot. Note that in the latter plot, the fact that both J1 and J2 are synchronous forces the instances of UOBs C and D to start and complete simultaneously.

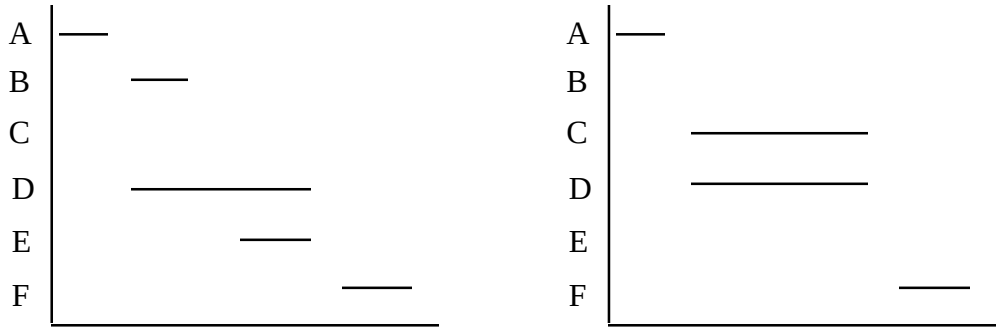

Figure 3-21
Activation Plots for Figure 3-20

Figure 3-22 is an example of a way to combine two different types of junctions to allow more freedom in the timing and sequencing of activations.

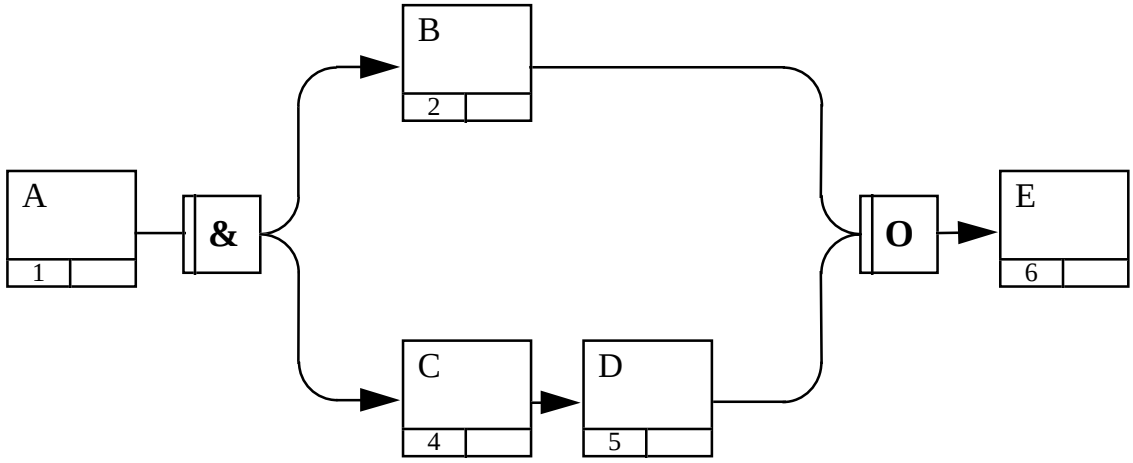

Figure 3-22
Fan-out AND Junction Followed by a Fan-in OR Junction

Although instances of UOBs B and C occur after an instance of A in activations of Figure 3-22, possible activations of the process are represented in which an instance of one or the other may not complete, or even initiate, before the activation «proceeds» through the fan-in OR junction and an instance of E occurs. Such activations are allowed because of the use of an asynchronous fan-in OR junction which governs the convergence of the two threads. For a successful process activation, although both threads must complete *at some time or other*, it is sufficient for only one of the threads to have completed prior to an instance of E. Figure 3-23 provides plots of three basic activation patterns permitted by Figure 3-22. The leftmost plot exhibits a pattern that would be permitted if the OR junction were an AND junction instead (or, equivalently, if the OR junction were synchronous). In the leftmost plot, instances of both B and D (hence also C) complete before an instance of E. In the middle plot, an instance of E begins before an

instance of B completes (or even starts), and in the right plot, an instance of E begins before an instance of D completes.

Figure 3-23
Activation Plots for Figure 3-22

Of course, additional constraints on activations could further narrow the class of possible activations; e.g., one could require that the instance of B in an activation always begin before the instance of E completes. This constraint rules out the activations characterized by the middle plot (in which the instance of B occurs after E completes).

Some Concrete Examples

The following examples give further illustrations of the constructs discussed in the preceding section. Figure 3-24 depicts a scenario in which the receipt of a proposal is followed by cost and technical evaluations. The evaluations must be completed prior to contract award. Because the junctions are asynchronous, no constraints are placed on the relative timing of the initiation and completion of the evaluations. They must simply follow the receipt of the proposal and precede the contract award.

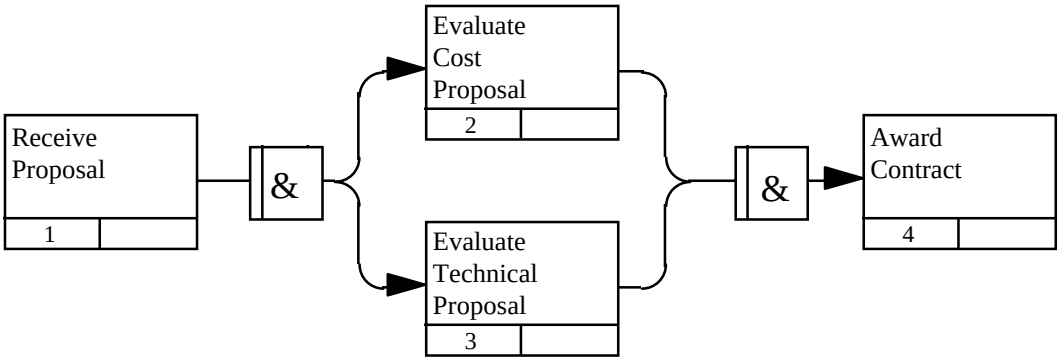

Figure 3-24
Asynchronous AND Junction Example

Contrast this with the scenario displayed in Figure 3-25 in which the synchronous AND describes a situation in which the cost and the technical evaluation *must* start simultaneously, but may end separately. If there had been an organizational rule that

required both to end together as well, Figure 3-25 would in addition have used a synchronous fan-in AND junction to describe the intended process.

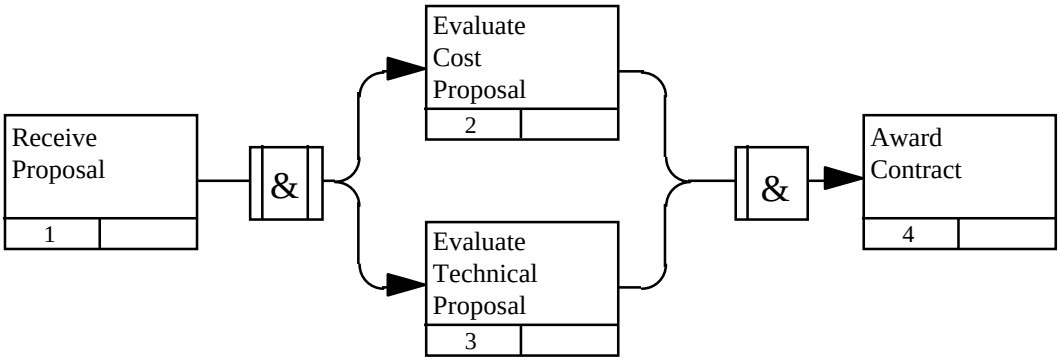

Figure 3-25
Synchronous AND Junction Example

Figure 3-26 shows a description of the *Select Contractor* scenario. This process description states that, following evaluation, one either rejects the proposal, or else accepts the proposal for core contract work, accepts the proposal for options to the contract, or both, before awarding the contract.

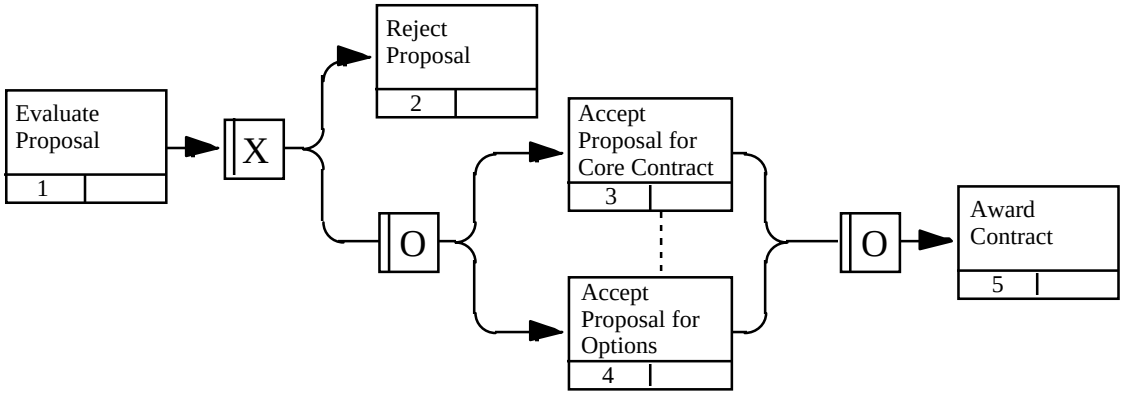

Figure 3-26
Asynchronous OR Junction Example

In the scenario depicted in Figure 3-26, *Reject Proposal* is a terminating activity; however, either of the other two activities (or both) will result in contract award. Note that a relational link indicates some relationship between the *Accept Proposal for Core Contract* and *Accept Proposal for Options* UOBs. Note also that this description is still partial in that it does not indicate what happens when the negotiations do not succeed. For example, in most situations, the contract award depends upon contractor acceptance of the terms of the funding agency, which may require the contractor to resubmit the proposal as a part of the negotiation process. Such information can be easily represented

in IDEF3 as additions to the current schematic or a decomposition of *Award Contract.* Note that there is nothing about the schematic that requires that the contract be awarded. The contract award would be forced only if a *constrained* precedence link (in the «left to right» direction) had been used to connect the fan-in OR junction with the *Award Contract* UOB.

Not all combinations of junctions represent genuine process logics. In particular, an XOR fan-out junction may not be followed by a fan-in AND junction in the fashion illustrated in Figure 3-27, since this would represent an inconsistent process, one that could not possibly be activated.

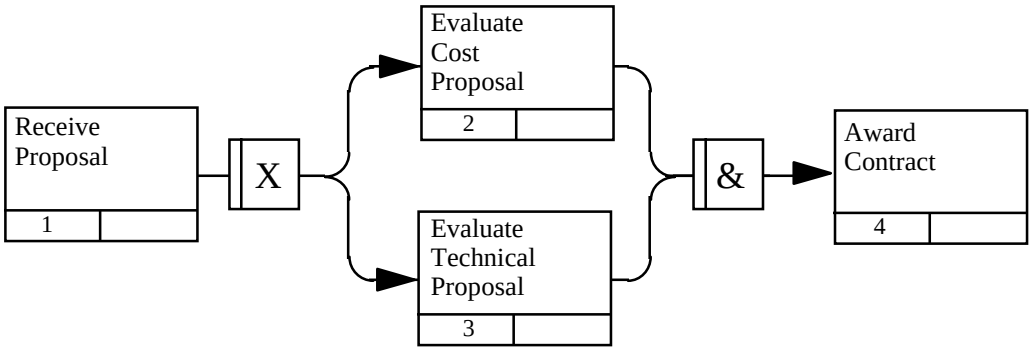

Figure 3-27
Invalid XOR/AND Structure Example

In Figure 3-27, after the *Receive Proposal* UOB, an XOR junction leads to two UOBs. This indicates that only one UOB—either *Evaluate Cost Proposal* or *Evaluate Technical Proposal*—will be realized on any given activation of the schematic. Consequently, the *Award Contract* UOB could never be realized because the requirement that both UOBs preceding the AND junction be realized in the same activation can never be met. Note that such schematics may nonetheless be useful, as the AS-IS process being captured may involve an undetected inconsistency. In the situation characterized in Figure 3-27, perhaps contracts were never awarded; thus, the IDEF3 schematic identified an organizational problem and enabled conflict resolution. This type of structure is never correct in a TO-BE description of some proposed system, organization structure, or process. In either case, however, the description validation process should identify structures of this type as IDEF3 schematic errors.

UOB Decompositions

Elaborations capture and structure detailed knowledge about processes. If the UOB represented by a box in a given schematic is highly complex, it may be useful to *decompose* the UOB explicitly into its component UOBs. The way this is represented in IDEF3 is that the original box is correlated with another IDEF3 schematic which

represents an «exploded» description of the UOB, providing a further level of descriptive detail about the UOB. This schematic is known as the *decomposition* of the original UOB box. Decompositions allow the user to capture descriptions at varying levels of abstraction. Decompositions enable users to apply the «divide and conquer» principle—a powerful mechanism for managing complexity. By applying this principle repeatedly, it is possible to structure a process description to any level of detail. Decomposition also provides the ability to model the same process from different knowledge sources or different points of view. This is possible because IDEF3 allows the same UOB to have a number of different decompositions, or «views.» This capability is also useful in domain situations where a given process involves multiple functional organizations.

As noted, a UOB decomposition is just another IDEF3 process schematic. In Figure 3-28, the use of decompositions is illustrated by an example from the domain of contracts management. The decomposed UOB box 3, which refers to the UOB *Receive and Activate Contract*, is called the *parent* UOB box. Where there is no danger of ambiguity (i.e., where no other box refers to the same UOB), the indicated UOB can also be called the parent UOB. Each decomposition of the parent box is a *child* decomposition. Each child decomposition is given a label and a unique number identifying it as one of potentially several decompositions of the parent UOB. The UOB boxes in a decomposition may have subsequent decompositions. (As seen in the figure, decompositions demand a special reference numbering scheme that is explained in the next section. For the moment, note that the rightmost digits in each UOB box is the UOB number. Moving to the left, the other two numbers in the UOB box provide additional information to the reader of an IDEF3 schematic.)

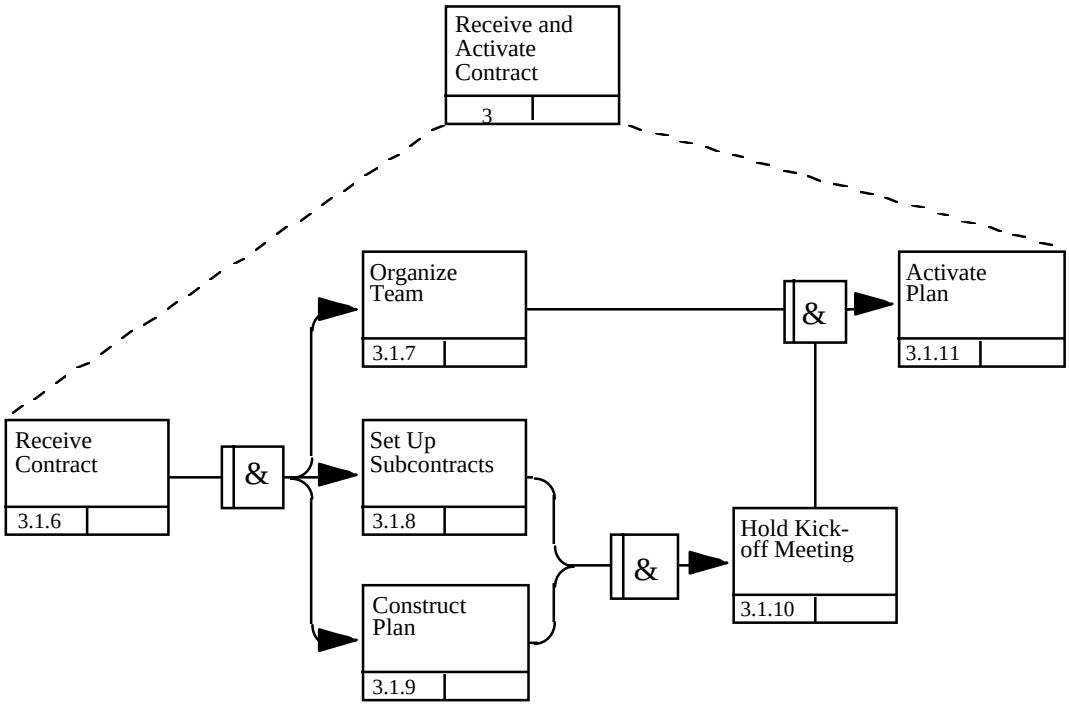

Figure 3-28
Decomposition 3.1 of the UOB *Receive and Activate Contract*

Multiple view decompositions may be consolidated into an *objective* view. The view presented in Figure 3-29 is an example of an objective view of the UOB *Hold Kick-off Meeting*. This is the view perceived by a neutral observer of the Kick-off Meeting process. However, the project manager of the contract will have a different perspective of this process; therefore, IDEF3 enables him to express his viewpoint via an alternative decomposition of the UOB. The project manager's decomposition of the UOB *Hold Kick-off Meeting* is shown in Figure 3-30.

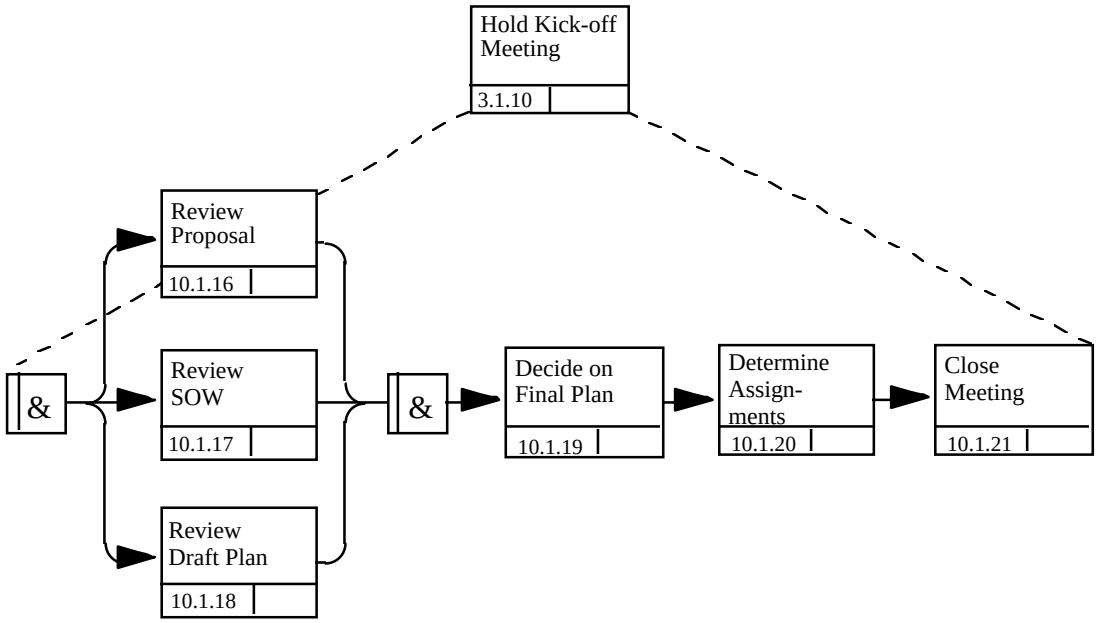

Figure 3-29
Decomposition 10.1 of *Hold Kick-off Meeting* UOB

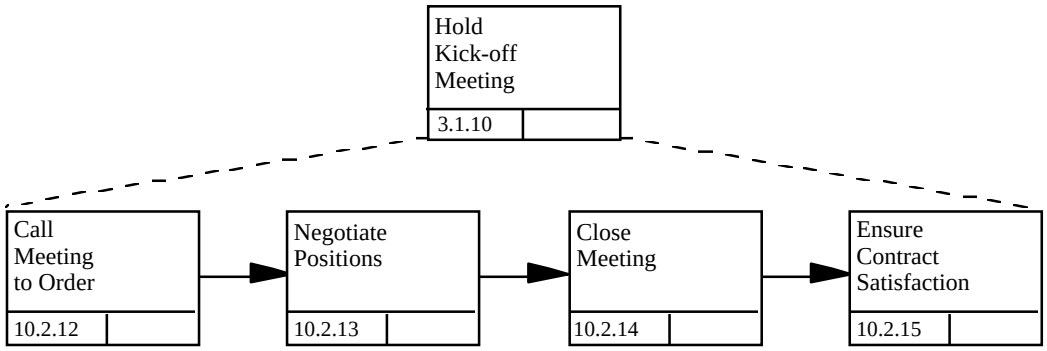

Figure 3-30
The Project Manger's View Decomposition

UOB Reference Numbering Scheme

A UOB box number is assigned to each UOB box in an IDEF3 Process Description. In general, however, a single IDEF3 description can be extremely complex, containing many UOB boxes, many of which can have multiple decompositions. In such schematics, the simple assignment of numbers to boxes, though sufficient for uniquely identifying each box, may not provide enough information. In particular, a single UOB box number conveys no contextual information about that UOB, i.e., information about where it fits in the overall process description. To provide this information, a more robust numbering

system may be used in IDEF3 schematics. These more informative designators are known as *reference numbers*. Specifically, at the top level in a hierarchy of decompositions, a box's UOB number and its reference number are identical. At lower levels of a decomposition, the reference number of a UOB box B consists of three distinct numerals separated by periods. The first number is the *last* number in the reference number of B's parent UOB. The second number is the number assigned to the particular decomposition of the parent box in which B occurs. (Numbers are generally assigned to decompositions and UOB boxes in order of creation, but this is arbitrary.) Finally, the third number in the reference is simply B's UOB box number. The reference numbering scheme thus displays a UOB box's UOB box number, the decomposition to which it belongs, and its parent UOB. The assignment of reference numbers is illustrated in Figure 3-31.

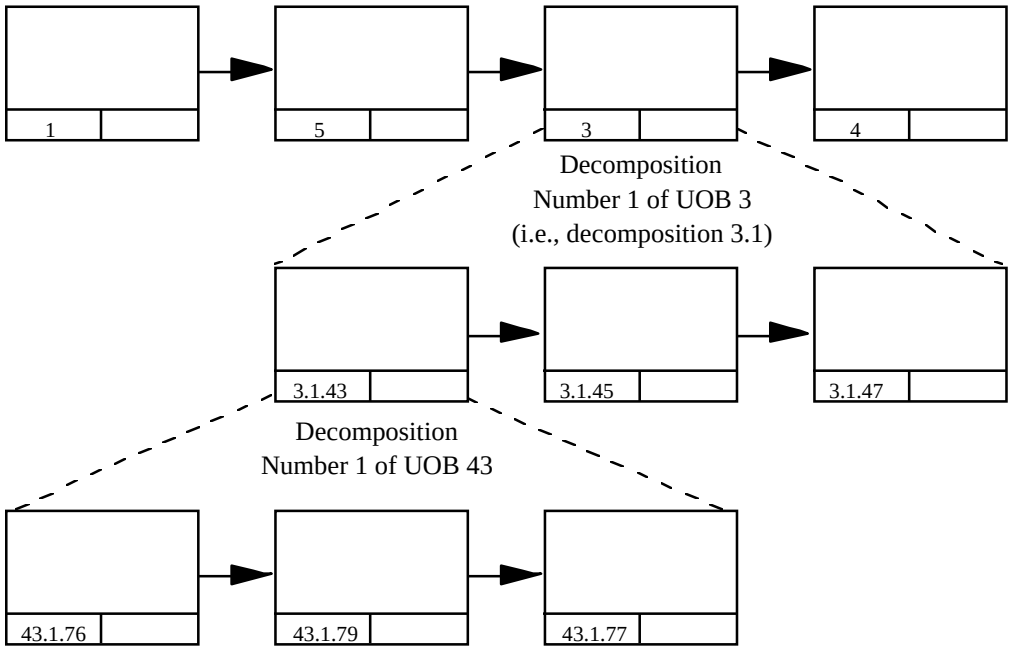

Figure 3-31
Unit of Behavior Numbering Scheme

If more than one person is involved in creating the description, constraints are enforced on the assignment of numbers to ensure that every UOB box is assigned unique box and reference numbers. The procedure suggested for UOB box number assignment is as follows. Each person is assigned a set of numbers (e.g., Joe gets 1-99, Jane gets 100-199, etc.), and can assign UOB box numbers only from his or her allocated set. Once the initial set of numbers is used, additional numbers can be assigned as necessary. By applying this number assignment procedure, the lead analyst in the development effort

can be assured that each UOB in the final combined description will contain unique box and reference numbers.[4]

Partial Descriptions

UOB boxes are joined together by links. Because of the description capture focus of IDEF3, it is possible to conceive of UOBs without links to other parts of an IDEF3 schematic, as the example in Figure 3-32 illustrates. These typically result early in the fact collection activity as references are made by the domain expert to the existence of events or activities without assertions being made about how they fit together.

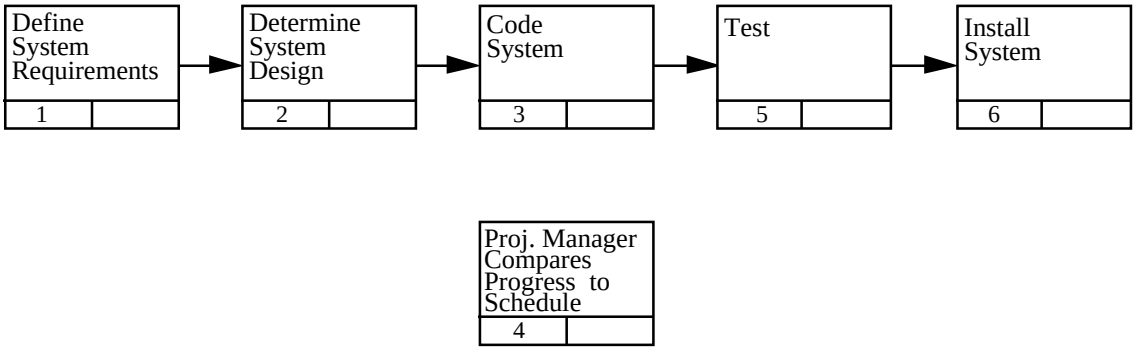

Figure 3-32
Disconnected UOB Example

In Figure 3-32, UOB 4 has no links to the rest of the schematic. This could either represent the actual situation or reflect the uncertainty of the domain expert's knowledge about the presence or absence of links. In this illustration, the schematic represents the actual situation. The concept that makes the *Project Manager Compares Progress to Schedule* UOB part of this schematic is the object *Project Schedule* shared by other UOBs in the schematic. The IDEF3 method, by allowing the creation of such stand-alone UOBs, facilitates the creation of *partial descriptions*. It allows users to represent the state of the world as they know it, with no enforced constraints on completeness. In fact, a common error that can be committed in the course of developing descriptions is to attempt to «drive to completion» inherently incomplete sets of descriptions.

---

[4] The UOB reference numbering scheme is provided to facilitate coordinated team effort and easy navigation across multiple views and varying levels of granularity in the description. The UOB reference numbering scheme is particularly important in a paper-based environment or one where the software tools being used provide limited integration support. For convenience in presentation, however, users may choose not to display UOB reference numbers or to number UOBs from left to right and from top to bottom as they appear in the schematic.

Referents

Referents enhance understanding, provide additional meaning, and simplify the construction (i.e., minimize clutter) of both process schematics and object schematics. Referents may be used in IDEF3 Process and object schematics to do the following.

1. Refer to a previously defined UOB without duplication of its definition to indicate that another instance of a previously defined UOB occurs at a specific point in the process (without loopback).

2. Transfer control or indicate a loopback in the processing.

3. Form references or links between the process schematics and object schematics.

The graphical symbols for the two basic styles of referents are displayed in Figure 3-33. Each type of referent may be used either in a process schematic or an object schematic, although process schematics tend to make more extensive use of the Call-and-Continue style referent. Using a Call-and-Continue referent indicates that the referenced element needs only to initiate before the focus IDEF3 element (that is, the IDEF3 element that makes the reference) can progress to completion. The use of a Call-and-Wait referent indicates that the referenced element needs to both initiate and complete before the focus IDEF3 element can progress to completion.

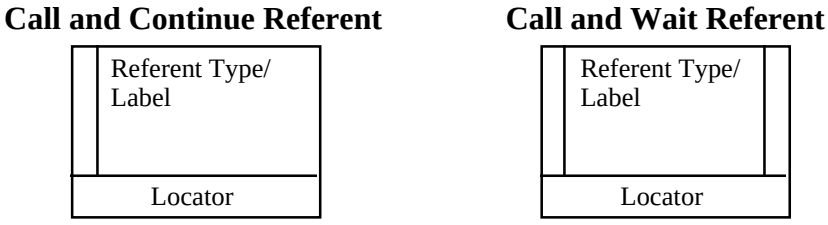

Figure 3-33
Referent Symbol Syntax

The type of thing signified by a referent—known, therefore, as the *referent type*—is indicated by prefixing one of the terms «UOB,» «SCENARIO,» «TS,» or «GO-TO,» followed by a slash, to a label for the thing signified (e.g., UOB/Perform Mission Area Analysis). Referents also include a field to note a locator for the thing signified. A summary of the referent types and referent labeling guidelines is provided in Figure 3-34.

Referent Type	Referenced Element Label	Locator
UOB	UOB Label	UOB#
SCENARIO	Scenario Label	Scenario #
TS	Transition Schematic Label	Transition Schematic #
GO-TO (used only in process schematics)	UOB Label Scenario Label Junction Type (i.e., &, O, or XOR)	UOB#/Scenario # or Decomposition # in which the ID occurs Scenario # Junction #/Scenario # or Decomposition # in which the ID occurs

Figure 3-34
Referent Symbol Structure

The following paragraphs summarize the semantics of the possible uses for referents in both the process and object schematics. To acquire a full understanding of the semantics associated with referents, readers need to become more familiar with object schematics. Readers may find it useful to read the following subsections which describe Call-and-Wait and Call-and-Continue referents, paying particular attention to their use in process schematics. Readers should then proceed to the discussion of object schematics. Once a basic understanding of object schematics is acquired, the reader may return to the Call-and-Continue and Call-and-Wait referents subsections to obtain a full understanding.

Call-and-Continue Referents

If the call-and-continue referent type is «UOB,» «SCENARIO,» or «GO-TO,» it may have no outgoing precedence link. To do otherwise would be inconsistent with the semantics of a precedence link. To understand why, one need only consider the semantics of call-and-continue referents and precedence arrows. A call-and-continue referent indicates that when an instance of the referenced UOB begins, for example, the process may continue. The precedence constraint, however, specifies that the process may continue only after an instance of the UOB both starts and completes. Hence, the two differing semantics cannot be applied simultaneously without violating the grammar.

If the referent type is «UOB,» the label must be a UOB label; this means that another instance of a previously defined UOB occurs at a specific point in the process (without loopback). If this referent type is attached to a transition arc in an object schematic, an activation of the referenced UOB must be initiated before the state transition is allowed (see referent discussion in the object schematics subsection). If this referent type is attached to an object state in an object schematic, it indicates that the referenced UOB sustains the object in the state. Similar semantics apply for Scenario-type referents attached to object states. See the subsection entitled, «Referents Attached to Object States,» for more detail.

If the referent type is «SCENARIO,» the label must be a Scenario label. If this referent type is used in a process schematic, it indicates that the next happening in the process flow is an occurrence of an activation of the referenced Scenario. That is, all decompositions of the named Scenario would be activated. If this referent type is attached to a transition arc in an object schematic, an activation of the referenced Scenario must start before the state transition is allowed (see referent discussion in object schematics subsection).

If the referent type is «TS,» the label must be a Transition Schematic label. If this referent type is used in a process schematic, it must be attached to a UOB with a simple connecting link (i.e., no precedence links). This use indicates that the referenced Transition Schematic must initiate sometime during an activation of the UOB. If this referent type is used in an object schematic, it must be attached to some point on a transition arc between states (i.e., it may not be attached to an object or object state). A Call-and-Continue TS referent attached to a transition arc between states indicates that the object must initiate a transition through the states of the referenced Transition Schematic before the state transition is allowed (see referent discussion in object schematics subsection).

If the referent type is «GO-TO» and it refers to a UOB, the next happening in the process is an occurrence of the referenced UOB. This type of referent is often used to document loops in a process. If the referent type is «GO-TO» and it refers to a Junction, the next happening in the process is an occurrence of the UOB(s) following the referenced junction. Go-to referents are always Call-and-Continue type referents.

Call-and-Wait Referents:

If the referent type is «UOB» or «SCENARIO,» it may have an outgoing precedence link. Call-and-wait «GO-TO» referents are not permitted.

If the referent type is «UOB,» the label must be a UOB label; this means that another instance of a previously-defined UOB occurs at a specific point in the process (without loopback). If this is attached to a transition arc in an object schematic, an activation of

the referenced UOB must be initiated and completed before the state transition is allowed (see referent discussion in object schematics subsection). If this referent type is attached to an object state in an object schematic, it indicates that the referenced UOB sustains the object in the state throughout its duration. Further, using a Call-and-Wait referent adds the constraint that the succeeding object state(s) may not be realized prior to completing the process represented by the UOB. Similar semantics apply for Scenario-type referents attached to object states. See the subsection entitled, «Referents Attached to Object States,» for more detail.

If the referent type is «SCENARIO,» the label must be a Scenario label. If the referent type «SCENARIO» is used in a process schematic, the next happening in the process flow is an occurrence of an activation of the referenced Scenario. That is, all decompositions of the named Scenario would be activated and completed before the next happening in the process flow. If a Scenario-type referent is attached to a transition arc in an object schematic, an activation of the referenced Scenario must complete before the state transition is allowed (see referent discussion in object schematics subsection).

If the referent type is «TS,» the label must be a Transition Schematic label. If this referent type is used in a process schematic, it must be attached to a UOB with a simple connecting link (i.e., no precedence links). This use indicates that the completion of the attached UOB is conditioned on an object transitioning through the referenced Transition Schematic. If this referent type is used in an object schematic, it must be attached to some point on a transition arc between states (i.e., it may not be attached to an object or object state). A Call-and-Wait TS referent attached to a transition arc between states indicates that the object must transition through the states of the referenced Transition Schematic before the state transition is allowed (see referent discussion in object schematics section below).

Using Referents in IDEF3 Process Schematics

In this section, the use of referents in process schematics is discussed. The use of referents in object schematics will be discussed in a later section after introducing the basic state transition schematics. This presentation strategy is intended to facilitate discussion about how to integrate the process-centered and object-centered views of a process.

Figure 3-35 is a process schematic depicting the requirements planning process. The referent in Figure 3-35 (a) indicates that a Transition Schematic (with Transition Schematic number 1) must be traversed before the *Prioritize Needs* UOB can initiate in an activation of the process. This construct reflects the notion that a Statement of Need (SON) traverses through a number of states that must be realized sometime during mission area analysis. Figure 3-35 (b) demonstrates the use of a Go-To referent to show the possibility of looping back to the *Perform Mission Area Analysis* UOB. The junction

referent in Figure 3-35 (c) indicates that the processing after the UOB *Explore Concept* is transferred to the junction J4 in decomposition 2.1. Referents may also be used to indicate that the situation represented by a UOB box in some other location is to be duplicated at some point. This use of a referent is illustrated in Figure 3-35(d). In the example, an instance of a process path traversing through the *Define Concept* UOB is followed by the duplication of the processing that occurs in the UOB *Perform Alternative Trade-offs* (with UOB number 15 and which is found in decomposition 9.1).

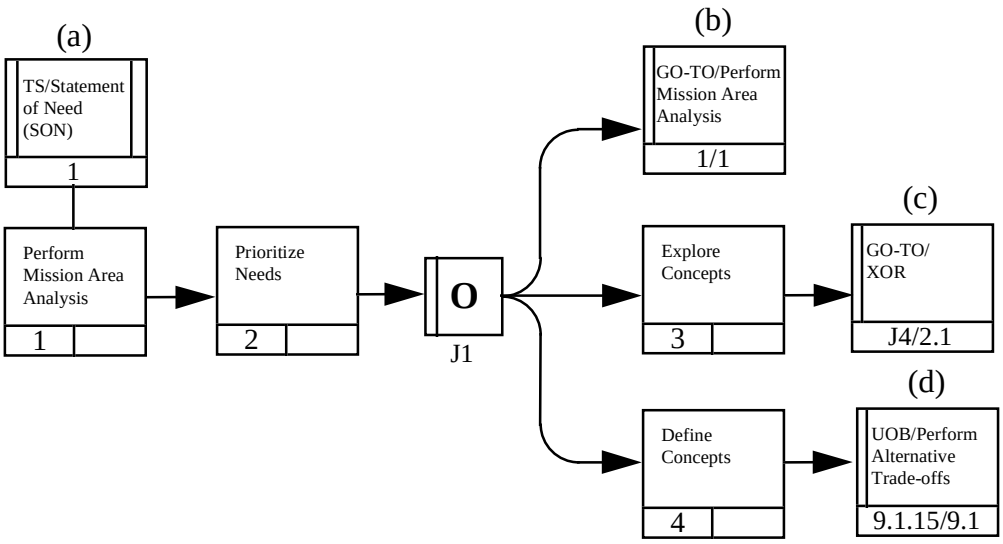

Figure 3-35
Process Schematic with Go-To Referents

Object Schematics

This subsection describes how to express detailed *object-centered* process information; that is, information about how objects of various kinds are transformed into other kinds of things through a process, or how objects of a given kind change states through a process. Thus, for example, a drive train, a chassis, and an auto body might be combined into a car. Again, in a given heating process, a quantity of water might change states from frozen, to cold, to warm, to hot, to boiling. An object-centered representation should, therefore, capture both various kinds of things, as well as kinds of things in various states.

Objects and Object States

An object of a certain *kind*, like a chassis, will be represented simply by a circle containing an appropriate label, as illustrated in Figure 3-36. These will be known as *kind* symbols.

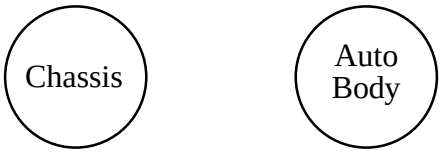

Figure 3-36
Kind Symbols

A certain kind of object being in a certain state will be represented by a circle with a label that captures the kind itself and a corresponding state, representing thereby the type, or class, of objects that are in that state (within a given process). For example, frozen water will be indicated by the label «Water:frozen», cold water by «Water:cold», and so on. These new constructs are called *object-state* symbols, and are illustrated in Figure 3-37.

Figure 3-37
Object-state Symbols

The construction of complex representations built from kind symbols and object state symbols are known as *object schematics*. The remainder of this section is devoted to the syntax and semantics of these constructs. For a lengthy and detailed discussion of the background to these constructs and to *ontology* modeling in general, see the *IDEF5 Method Report* (KBSI 1994).

3.3.2 Transition Schematics

The first and most basic construct is the basic *state transition schematic* (or simply, *transition schematic*, for short) shown in Figure 3-38.

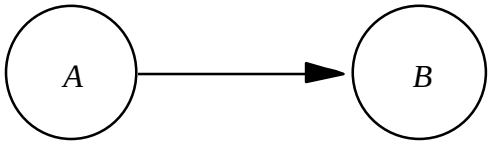

Figure 3-38
Basic State Transition Schematic

Intuitively, a basic transition schematic signifies a certain pattern of events, a certain type of situation that can occur, namely, a situation in which there is an object *a* in a given state *A* followed by an object *b* in state *B*. Typically the object *a* is, over a certain time

modified, transformed, or consumed to yield *b*. More often than not, *a* and *b* will be the same object, such as a quantity of water that transitions from a solid to a liquid state, or a given car body changing from an unpainted to a painted state. However, this is not always so; for instance, an incineration process might involve a state transition in which a piece of wood is consumed, yielding a pile of ashes. Hence, for the sake of generality, the default semantics of a basic transition schematic is the weaker of the two readings. Should one wish to express the stronger reading explicitly in which one and the same object undergoes the state transition, one can use a double headed arrow, as shown in Figure 3-39.[5]

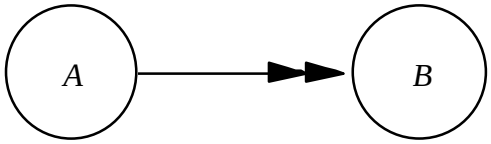

Figure 3-39
Basic State Transition Schematic with a *Strong* Transition Link

Just as the transition from *A* to *B* typically involves the same object, it is also typically the case that the object in state *A ceases to be in A* prior to its transition to *B*. Thus, a quantity of water transitions from solid to liquid; a car body transitions from unpainted to painted; and so on. However, this needn't always be the case. For example, a room with (at least) one person in it might transition to a room with (at least) ten. That is, a room with at least ten people in it is also a room with at least one person in it.

In general, then, the semantics of a basic state transition schematic is that, in an occurrence of the indicated transition, there is first an object *a* in state *A*, and *subsequently* an object *b* that comes to be in state *B*; that is, it is required that *a* be in state *A* before *b* comes to be in state *B*. It is *permitted*, though perhaps not typical, that the object in state *A* be distinct from the object that comes to be in state *B*; and it is *permitted*, though perhaps not typical, that *a* remain in state *A* after *b* comes to be in state *B*.

It is important to note that, despite having roughly the same appearance, the semantics of a solid-tipped arrow in an object schematic is different than the semantics of an arrow in a process schematic. In process schematics, an arrow implies full temporal precedence: an instance of the UOB indicated at the tail of the arrow must *complete* no later than the point at which an instance of the UOB indicated at the head of the arrow begins. By contrast, in an object schematic, the arrow implies precedence only with

[5] Identity may be in terms of chemical structure, mass, physical form, function, etc. For example, grape juice becomes wine after undergoing a fermentation process. One might argue that the «stuff of» the kind grape juice is the same as that of the resulting kind wine. Other people having different attunements may perceive the two kinds as being entirely different based on, for example, chemical composition of the two kinds. We recommend that the assumed criteria for identity be established or characterized when there is possible ambiguity.

regard to starting points: the object in the state indicated at that tail of the arrow must *begin* to be in that state before the transition to an object in the state indicated at the head of the arrow. The reason for the switch to this weaker sort of precedence in state transition schematics is noted above: a transition only involves a change from an object in one state to an object (possibly the same object, possibly different) in another; though it may *typically* be so, the object in the initial state of the transition *needn't* cease being in that state after the transition. To allow for this type of transition, the weaker semantics is used for the arrow in object transition schematics.

Conditions

It is important to distinguish between the characterization of an object of a given kind (or in a given state) and the conditions or rules that govern how the object comes to be of another kind (i.e., how it transitions to and from that state). (Henceforth, explicit reference to kinds will be omitted, as states are the central focus of this section). Four general classes of conditions are distinguished in IDEF3: entry, transition, state, and exit. State and exit conditions are associated intrinsically with states, while entry and transition conditions are associated with the interface between states and transition links as depicted in Figure 3-40. Consequently, the former two are listed in the elaboration for an object state, while the latter two are listed in the elaborations for transition links. (Figure 3-40 is not illustrating new graphical elements for object schematics! The purpose of the figure is to picture where each type of condition is applicable in an object schematic.)

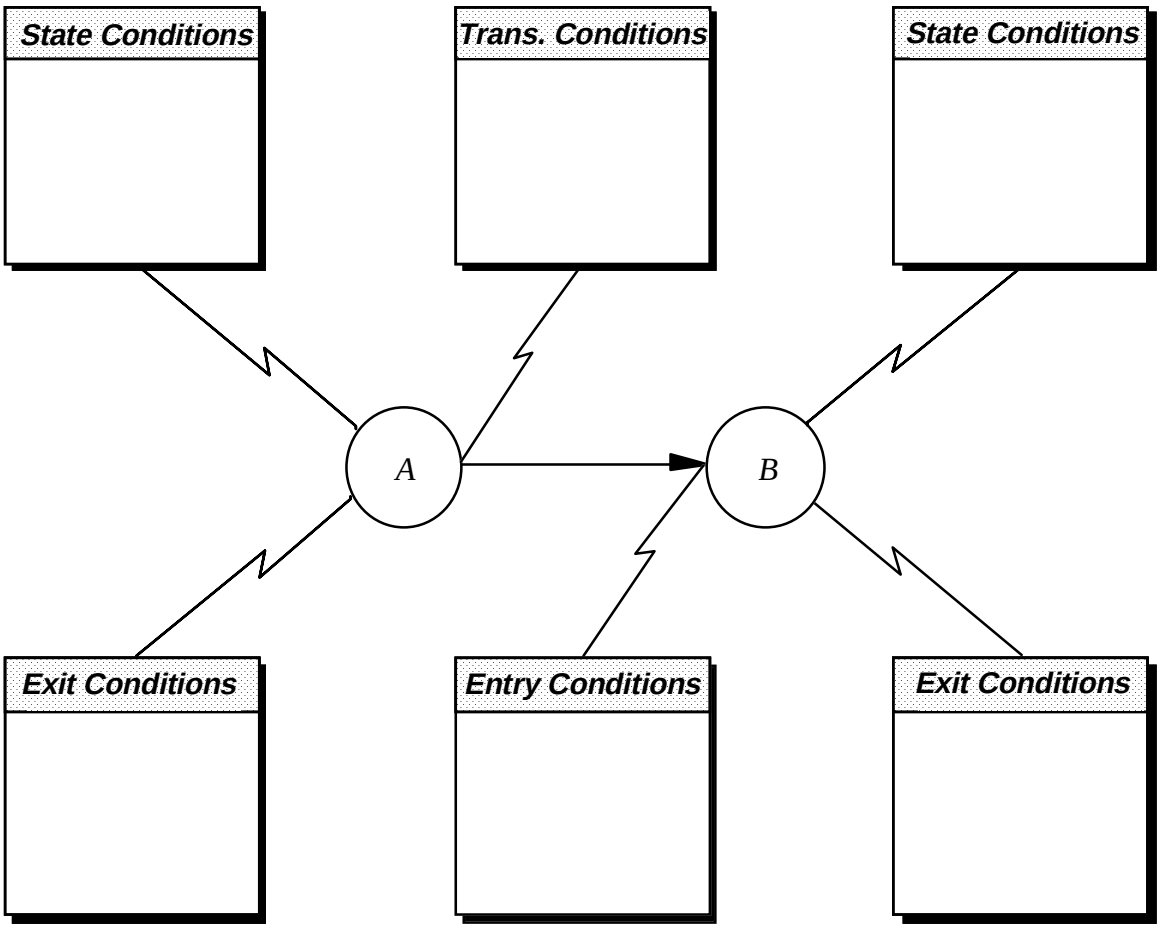

Figure 3-40
Object Schematic Conditions

State conditions are those necessary for an object to be in the state in question. For example, to be in a frozen state (at sea level), water must satisfy the condition of being at or below 32 degrees Fahrenheit (though *other* conditions—e.g., salinity below a certain degree—may be required for that condition to be *sufficient* for water to be frozen). Note that this information is independent of *how* some quantity of water might come to be in that state. Exit conditions are simply conditions sufficient for an object in a given state to cease being in (i.e., to exit) that state. For instance, water ceases to be in a frozen state if heated to a temperature above 32 degrees Fahrenheit.

Notice that there is no implication that it is known what state, if any, an object in a given state S transitions to upon satisfying an exit condition for S; this is the essential difference between an exit condition and a transition condition. Transition conditions apply to the «interface» between a state and an outgoing link and consist of conditions that are individually necessary and jointly sufficient for there to be a transition (or, at least, an *attempted* transition) of an object in a given state (A in Figure 3-40) to a

(possibly different) object in the destination state of the relevant link (B in Figure 3-40). Finally, entry conditions apply to the interface between a state and an incoming link and consist of conditions that are sufficient for an object to enter that state given a (possibly different) object in the source state of that link that has met the relevant transition conditions.

Note that for any given transition from one state to another in an object schematic, there is no requirement for any determinate or identified conditions of any of the four types. In relatively simple schematics, for example, the semantics will be evident from the labels on the state and UOB symbols.

Using Referents in IDEF3 Object Schematics

As with process schematics, the finer details of a state transition are left to the elaborations of the object states. However, by attaching *referents* to arcs one can add useful additional information about a state transition explicitly to a corresponding schematic.

Referents Attached to Transition Links

Referents used in object schematics can signify either a UOB, a scenario, or a transition schematic. Intuitively, if the referent is a UOB or Scenario referent, the referent signifies the process during which the indicated transition occurs, or at least a process involved in the transition. On the other hand, if the referent is a transition schematic, the referent indicates that the transition—from an object in state *A* to an object in state *B*, in Figure 3-41—involves transitions through the intermediate states signified in the indicated transition schematic.

The syntax for the most typical case—a single referent attached to a transition arc in a basic transition schematic—is illustrated in Figure 3-41.

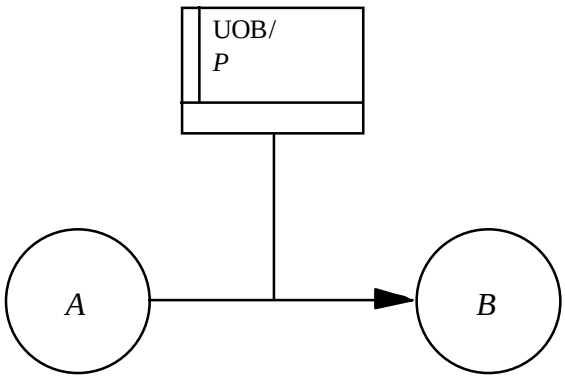

Figure 3-41
Basic Transition Schematic with UOB Referent

Typically, *P* will be the process during which the indicated transition occurs. Thus, in typical occurrences of the indicated process, there will be an object *a* in state *A* at the beginning of an instance *p* of *P*, and subsequently an object *b* at some point after the beginning of *P*. However, as noted, the referent in Figure 3-41 might indicate only a process *involved* in the transition from state *A* to state *B*. Thus, the general semantics of Figure 3-41 requires only that, in an occurrence of the indicated transition, there must be an object in state *A prior to* or *at the start of* an instance of *P*.

This semantics of transition schematics can be presented in terms of «interval diagrams»—as seen in Figure 3-42—that illustrate the temporal relationships between the various situations that occur in an instance of the pattern of events represented by an object schematic. Each horizontal line in an interval diagram represents the time interval over which a given UOB or scenario occurs, or over which a particular object is in a given state. A vertical line represents the starting or ending point of an interval. «*Aa*» is short for «*a* is in state *A*», and likewise for «*Bb*». These diagrams are useful because even a basic state transition schematic permits multiple «instantiation patterns,» multiple ways that real world events can count as instances of the schematic. Thus, all of the interval diagrams in Figure 3-42 depict legitimate instantiation patterns for the schematic in Figure 3-41. Interval diagram 1 shows a case in which there is an object *a* in state *A* prior to the beginning of an instance *p* of *P*, and in which the object *b* to which there is a transition continues in that state until after the end of *P*. Interval diagram 2 indicates a state transition of an object *a* from *A* to *B* that is *instantaneous* (relative to some time grain). Interval diagram 3 indicates two important possibilities. First, it illustrates that *p* could begin simultaneously with (but not prior to) *a*'s coming to be in state *A*. Second, it illustrates that *b*'s coming to be in state *B* might occur before *a* ceases to be in state *A*. Typically, of course, in such a case *a* and *b* will be distinct objects; *a*'s being in state *A* might be a precondition for *b*'s coming to be in state *B* during *p*, as, for example, a certain circuit (*a*) being open (*A*) might be a precondition for a certain warning light (*b*) to activate (*B*). Finally, interval diagram 4 indicates a case in which *a* ceases to be in state *A* prior to the start of *p*, and then comes to be in state *B* after *p* ends.

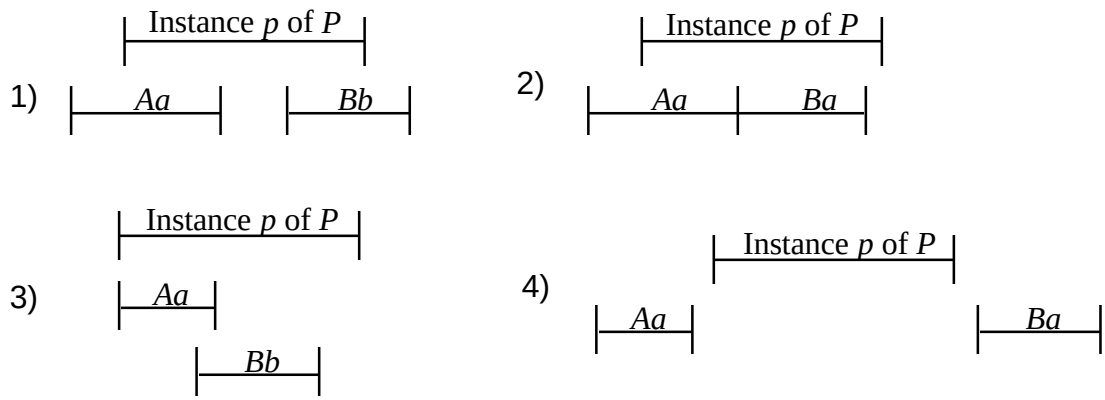

Figure 3-42
Interval Diagrams Representing Instances of Figure 3-41

Because the referent in a basic transition schematic typically indicates the process by which the indicated transition occurs, cases with the structure depicted in interval diagram 4 might seem unwarranted. But again, the referent in Figure 3-41 need indicate only a process *involved* in the transition and not necessarily the complete transitioning process. For example, suppose that *A* is the state *water:frozen* and *B* is the state *water:gaseous* and *P* is a heating process (involving a heating element); but suppose in addition that in the indicated process, a block of ice is allowed to melt naturally and is only then heated by *P* which is operative only until the water boils, at which point the heating process is ended and the hot water is just allowed to transition into a gas naturally by evaporation.

The point of this weaker semantics is that IDEF3 is, among other things, a *process* (and *state transition*) *description capture* method. When describing a certain transition, one simply may not know what the full transition process involves, and in particular may know only about some intermediate process in the transition. The given semantics allows such a possibility.

It is often as important to understand what is ruled out of the semantics of a given representation as it is to understand what is permitted. Essentially, the only thing that can rule out a given course of events is the ordering of the starting points of its constituent situations. Thus, for instance, the two interval diagrams in Figure 3-43 do not depict legitimate instantiation patterns for Figure 3-42. Specifically, as in the first case of Figure 3-43, *b* comes to be in state *B* before the instance *p* of *P* begins, and, in the second case, *p* begins without *a* being in state *A*.

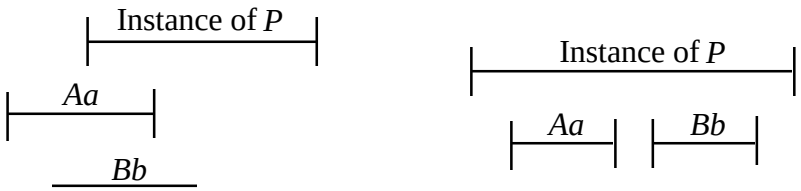

Figure 3-43
Patterns Excluded by the Semantics of Figure 3-41

This semantics holds regardless of whether the referent is a UOB or a Scenario referent; this is logical, since every scenario can be thought of as a finer-grained decomposition of a UOB. Matters are more or less the same if the referent is a TS referent. Consider the schematic in Figure 3-44, and suppose the referent refers to the schematic in Figure 3-41.

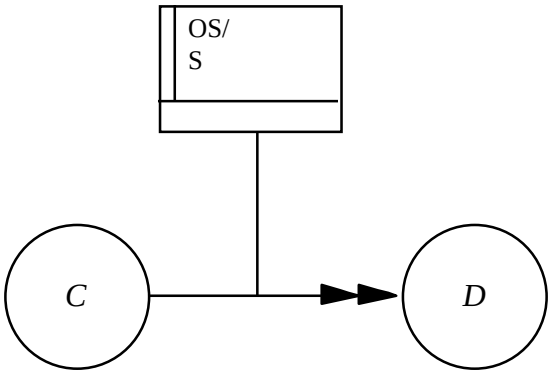

Figure 3-44
Transition Schematic with a Call-and-Continue Referent

This schematic signifies that, in an instance of the indicated transition, there is first an object *c* in state *C*, at which point an instance of Figure 3-41 begins, and hence some object *a* begins to transition to state *B* through an instance of the process *P*; *c* then transitions to state *D* at any point after the transition from *A* to *B* begins. The analyst determines exactly what it means for a transition to have begun in a given case. The main points here are that (1) the series of transitions referred by a TS referent must in some sense begin before the transition from *C* to *D* completes, and (2) the transition from *C* to *D* can occur regardless of whether or not that series of transitions has completed.

Additional information about the temporal sequencing of the events involved in a state transition can be added with a Call-and-Wait referent. Such a referent differs graphically from a Call-and-Continue referent by the addition of a second vertical line to the right of the referent name, as shown in Figure 3-45.

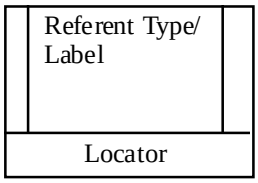

Figure 3-45
Call-and-Wait Referent Syntax

A Call-and-Wait referent indicates that the called situation must terminate before the next situation in the transition can transpire. Thus, again, with respect to Figure 3-41, an instance *p* of the UOB *P* would have to terminate before the object in state *A* could transition to state *B*. Note that the end of *p* could *coincide* with the completion of the transition in question. This is not implied in a state transition schematic with a Call-and-Continue referent. Rather, the process indicated by the referent must only *start* before the transition is completed; it *may* complete before the transition, but it could also legitimately continue well past the point of transition. Similarly, if the Call-and-Wait referent in question is an TS referent, then the referenced series of transitions must complete before the transition indicated in the schematic completes. Thus, if the TS referent in Figure 3-44 was a Call-and-Wait and was again referring to the schematic in Figure 3-41, then there would have to be a complete instance of a transition from *A* to *B* before an object *c* could complete a transition from *C* to *D*.

Referents Attached to Object States

It is not uncommon for a given situation to «sustain» an object in a given state; a refrigeration process, for example, might sustain a given substance in a solid state. Situations of this type can be represented by the construct in Figure 3-46.

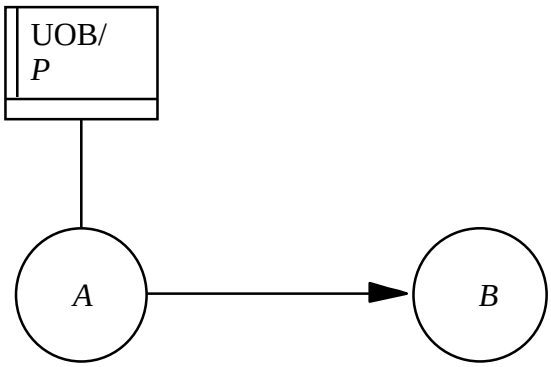

Figure 3-46
Sustaining an Object in a State

More generally, in an occurrence of Figure 3-46, there is an instance *p* of the UOB *P* and an object *a* in state *A* throughout the duration of *p*. This requires that such an *a* must exist when *p* begins. However, *a* could be in state *A* prior to the start of *p*; that is, it could

be *brought into* state *A* by some other process prior to *p* (the substance noted above might actually *become* solid through some sort of chemical reaction), and then *sustained* in that state by *p*. Thus, the two instantiation patterns in Figure 3-47 are both compatible with Figure 3-46.

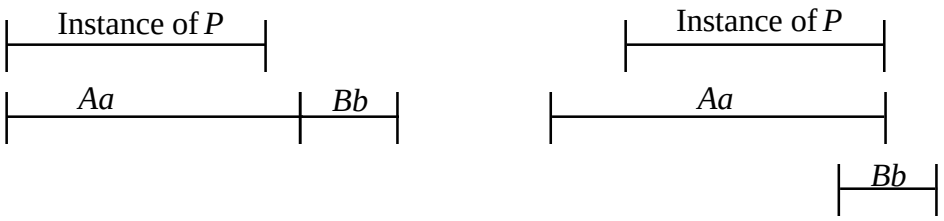

Figure 3-47
Instantiation Patterns for Figure 3-46

Note that in the right diagram, *b*'s coming to be in state *B* prior to the end of the instance *p* of *P* could be ruled out by changing the Call-and-Continue referent in Figure 3-46 to a Call-and-Wait. In that case, only the left diagram would represent a legitimate instantiation pattern.

Object Schematics with Multiple Referents

Often a more complex course of events than can be indicated by a single referent is involved in the transition from one state to another. The details of such a course of events, as one would expect, can be provided by a separate process schematic. However, it is useful to be able to represent that course of events explicitly in an object schematic. For this purpose, multiple referents can be attached to a single arc.

To interpret such schematics, think of the arc in a basic state transition schematic as a rough time line signifying the period over which the indicated transition occurs. Thus, the position at which a referent attaches to the line in a state transition schematic D signifies the relative temporal order in which the indicated UOBs, scenarios, or series of transitions begin in an instance of the transition signified by D. So, for instance, in Figure 3-48, a transition from *A* to *B* involves, first an instance *p* of the UOB *P* followed by an instance *q* of *Q*. Since the first referent is a Call-and-Wait, *p* must complete before *q* begins.

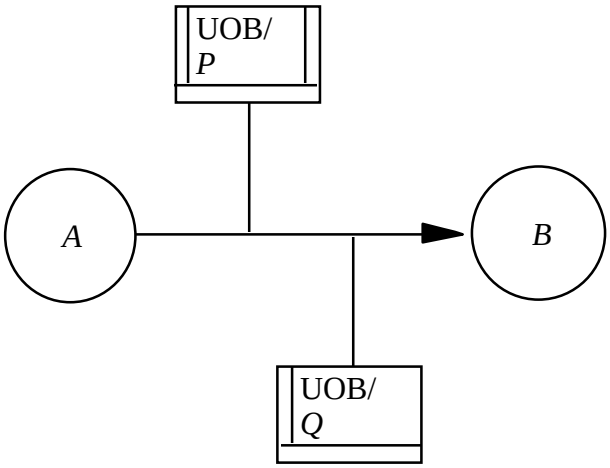

Figure 3-48
Object Schematic with Multiple, Temporally Ordered Referents

Figure 3-49 signifies a transition in which instances *p* and *q* of two UOBs begin simultaneously. Note that since the first referent is once again a Call-and-Wait, *p* must complete before the transition to *B* completes; since the second referent is a Call-and-Continue, the same does not hold for *q*.

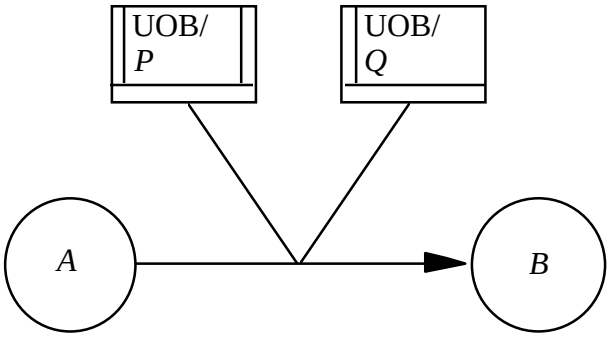

Figure 3-49
Object Schematic with Multiple Temporally Simultaneous Referents

Finally, Figure 3-50 illustrates the use of an additional symbol—a temporal indeterminacy marker (indicated by the small circle on the transition link)—to represent a transition in which there is (as far as is known) no definite temporal ordering to the UOBs involved in a transition. Instances of *P, Q,* and *R* are known to be involved in the indicated transition, but in any instance of the transition they can occur in any order relative to one another.

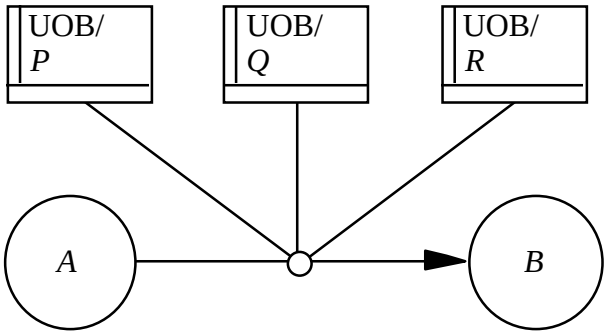

Figure 3-50
Object Schematic with Temporally Indefinite Referents

Because there is no definite temporal ordering between the indicated UOBs, only ordinary (i.e., Call-and-Continue) referents are used; Call-and-Wait referents would have no clear meaning.

Complex Transition Schematics

The processes that one might use to describe or model from an object-centered point of view are often too complex to be captured adequately by a basic transition schematic. Hence, it is possible to build complex transition schematics, i.e., schematics with multiple object state symbols. Complex transition schematics correspond, roughly, to complex process schematics. This supports the central role of transition schematics, providing object-centered views of processes. Hence, process schematics and transition schematics should be structurally similar. However, transition schematics are only a subclass of the entire class of object schematics that can be constructed in IDEF3.

Consider first the complex transition schematic, illustrated in Figure 3-51.

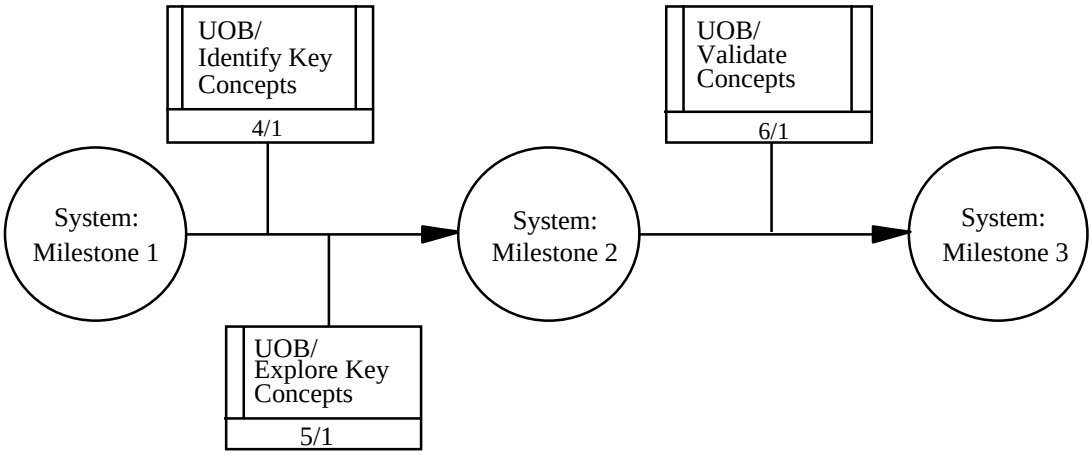

Figure 3-51
Complex Transition Schematic

The process described in this transition schematic involves a kind of system (called simply *system*) that transitions through three states: *Milestone 1*, *Milestone 2*, and *Milestone 3*. Because the first referent is a Call-and-Wait, in order for the system to transition from Milestone 1 to Milestone 2, the UOB *Identify Key Concepts* indicated by the referent must complete. The UOB *Explore Key Concepts* must then start, but because the referent in question is not a Call-and-Wait, it need not complete before the transition to Milestone 2; for instance, it may be sufficient for transition that *most* of the identified key concepts be explored. The UOB *Validate Concepts* must then begin after the transition to Milestone 2 and subsequently finish before, or at least no later than, the point at which the system has successfully transitioned to Milestone 3. Note that it is only *relative* placement on a transition arc that is important; the *distance* between two points of attachment is irrelevant, *unless* that distance is zero, i.e., unless two referents are attached at the same point, which signifies that instances of the indicated events are to begin simultaneously.

As with process schematics, a transition schematic is a structural whole; it describes, in a general way, the structure of the state transitions (or, at least, a prominent set of the state transitions) that one or more of the objects involved in a complex process undergo. Thus, a transition schematic cannot, in general, be broken into smaller pieces without losing information, as a transition schematic in general depicts an entire *series*, or *network*, of state transitions. The two basic transition schematics in Figure 3-52, for instance, do not carry as much information as the schematic in Figure 3-51.

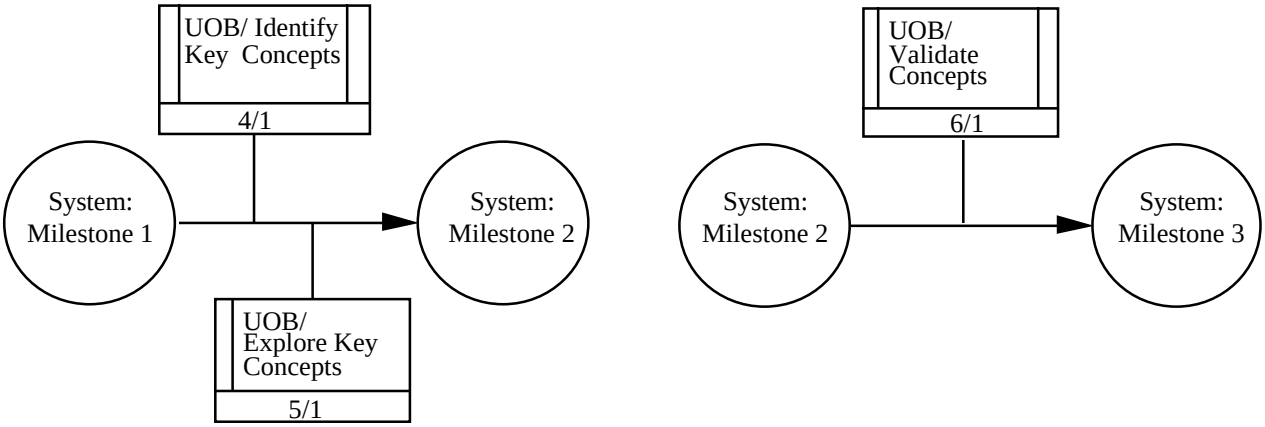

Figure 3-52
Transition Schematics Not Jointly Equivalent to Figure 3-51

These two schematics simply document the individual transitions from Milestone 1 to Milestone 2, and from Milestone 2 to Milestone 3. For all these schematics,these two transitions might never be successive in the system in any given instance. Because there is no implication in the semantics for basic transition schematics that Milestone 1 and Milestone 2 are mutually exclusive states (regardless of whether they actually are), these two figures are compatible with a situation in which there is a transition from Milestone 2 to Milestone 3 before a transition from Milestone 1 to Milestone 2, as depicted in Figure 3-53 (where «*M1(s)*» means the system is in the state Milestone 1, and so on):

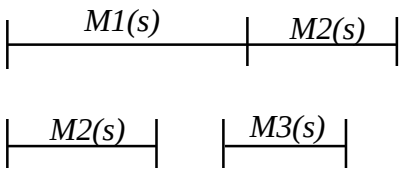

Figure 3-53
Possible Instantiation Pattern for the Schematics in Figure 3-52

Transition Junctions

Transition junctions provide a mechanism to specify the logic of potentially multiple paths in state transition behavior. For example, Figure 3-54 illustrates the use of an *inclusive disjunctive* junction, indicating that an object state may transition alternatively to one of a number of other states.

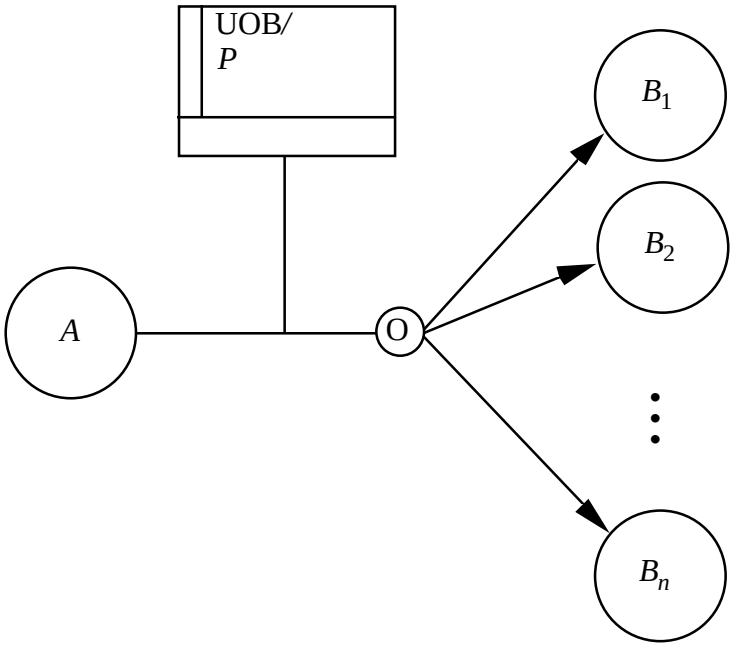

Figure 3-54
Disjunctive State Transition Schematic

Using a disjunctive junction that is *inclusive* permits a transition from **A** to one or possibly more than one of the subsequent states. To indicate *exclusive* disjunction, which permits transition to no more than one of the subsequent states, the construct in Figure 3-55 is used.

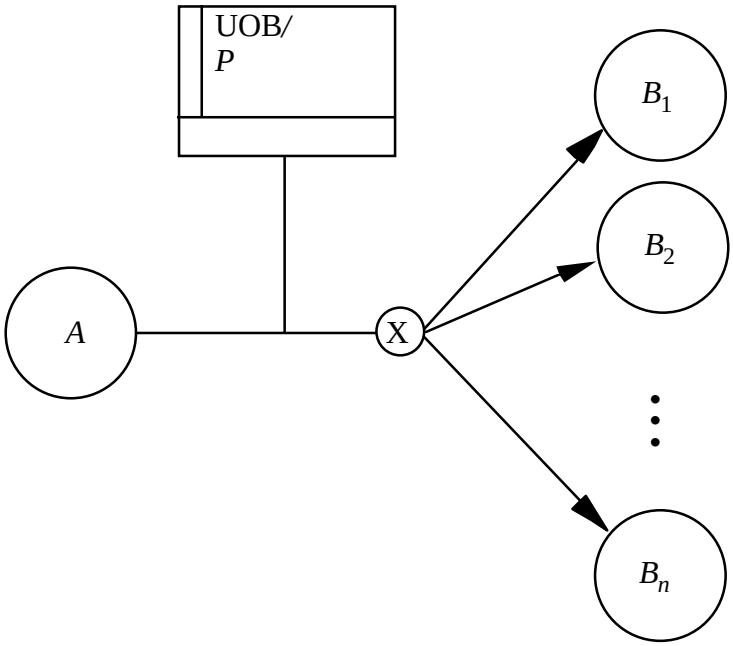

Figure 3-55
Exclusive Disjunctive State Transition Schematic

By the same token, a *conjunctive* schematic is introduced to indicate a transition from a given state to *all* of several subsequent states, as illustrated in Figure 3-56.

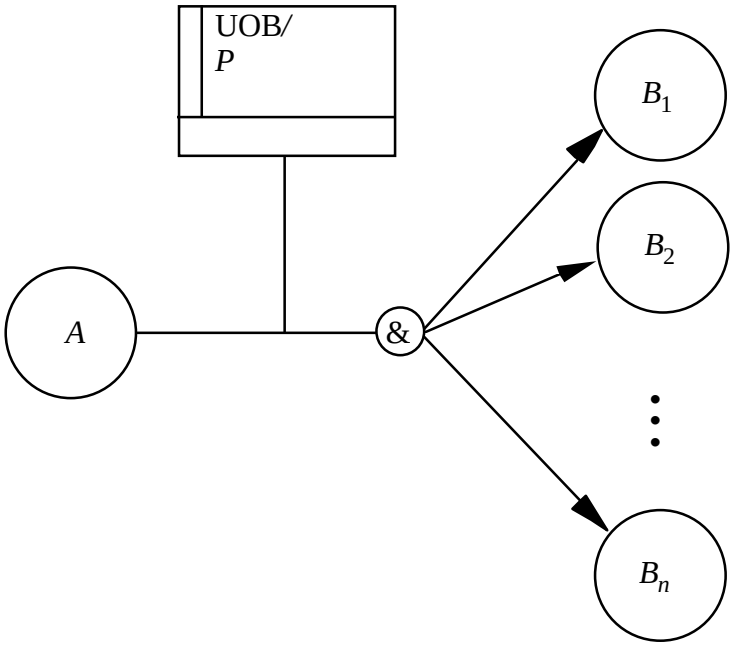

Figure 3-56
Conjunctive State Transition Schematic

The semantics for schematics including logical junctions is a generalization of the semantics for basic transition schematics. Intuitively, once again, the schematic in Figure 3-56 represents a type of situation in which there is a transition involving the process *P*, from an object *a* in state *A* to objects $a_1, \ldots, a_n$ in states $A_1, \ldots, A_n$, respectively. As with basic transition schematics, the object *a* must be in state *A* prior to, or at least no later than, the start of *p*; and b_i must be in state B_i or must begin to be in state B_i after the start of *p*. The possible variability of starting and ending points for the b_i is indicated by the use of dotted lines in the general instantiation pattern pictured in Figure 3-57.

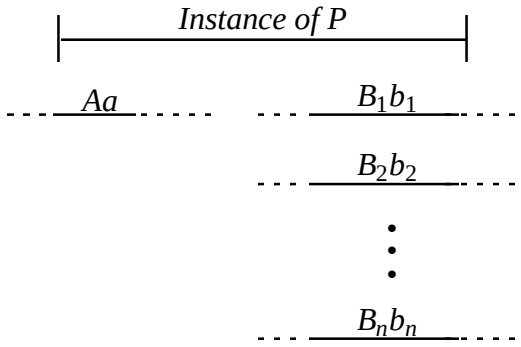

Figure 3-57
General Semantics of Figure 3-56

These logical schematics also have «converses»—specifically, where * is **O**, **X**, or **&**, Figure 3-58 is also a schematic.

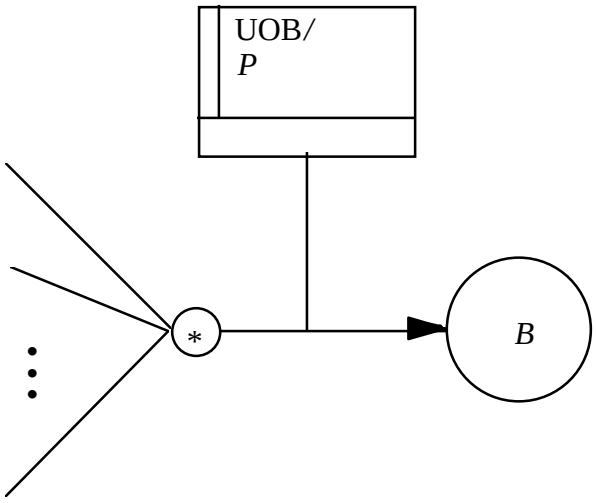

Figure 3-58
Converse Schematic

The semantics in each case will be exactly the converse of the corresponding schematic above.

Finally, it is possible that a transition can involve complex logic at both the beginning and end of the overarching process. For instance, it might be that objects in states A_1 *and* A_2 can transition either to state B_1 *or* B_2 in the course of a process *P*. The general syntax for characterizing such transitions is depicted in Figure 3-59, where * and # are any two of the logical symbols **O**, **X**, or **&** (possibly the same symbol).

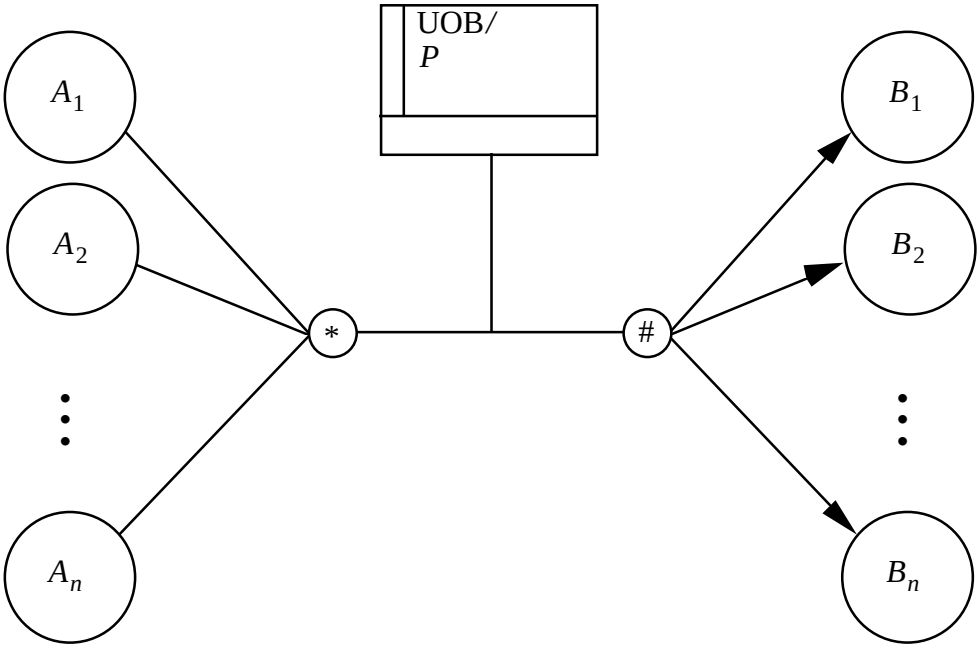

Figure 3-59
Using Multiple Junction Symbols to Display Complex Transition Logic

Schematics of this sort are generally ambiguous; for instance, letting $n = m = 2$, if * is & and # is O, Figure 3-59 could mean that, in an instance of P, a_1 and a_2 objects in states A_1 and A_2 can *both* transition to either B_1 or B_2, or that a_1 transitions to state B_1 and a_2 to B_2, and so on. Such ambiguities can be resolved in the elaboration form for the link.

Hiding Object State Information

As with composition and classification schematics, it is possible to hide information in object schematics. That is, for certain purposes, it may often prove useful to collapse complex state transition information about a given object into a single object state. For example, a series of state transitions involved in the process of heating water from freezing to boiling is depicted in Figure 3-60.

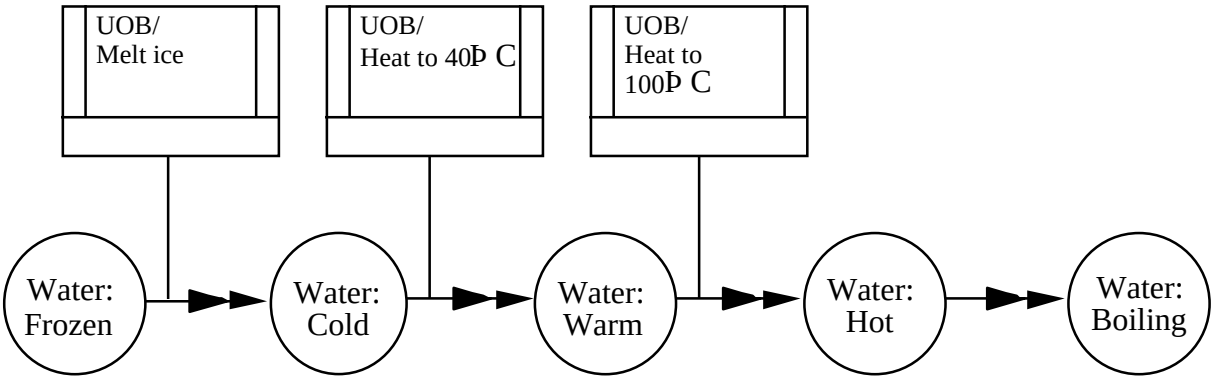

Figure 3-60
Object Transitions in a Heating Process

If, from a certain perspective, the intermediate transitions from ice to boiling water are irrelevant, then these transitions can be hidden in a single state in which the only relevant state is the coarse-grained *Water being heated* as depicted in Figure 3-61. Again a double circle is used; in this case an 'S' indicates that the type of information hidden is state transition information:

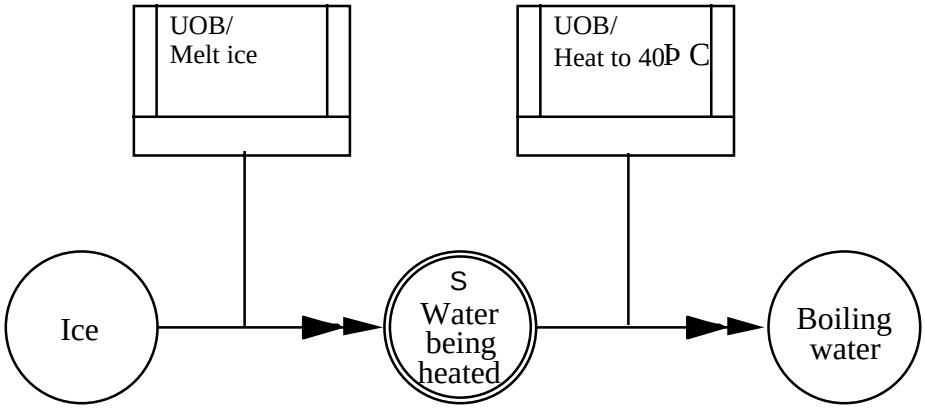

Figure 3-61
Hiding State Transition Information

The procedure for generating a coarse-grained schematic from a finer-grained schematic is not *quite* algorithmic. In the example, the state symbol for *Water being heated* can be thought of as directly replacing the «schematic» of Figure 3-60, consisting of the middle three kind symbols and their connecting links. However, the instantaneous transition schematic in Figure 3-60 had to be replaced by an ordinary state transition schematic, and an appropriate label had to be found for the attached process box. The exact nature of this alteration had to be determined by the nature of the represented process, and is, in general, a nonalgorithmic modeling decision.

Enhanced Transition Schematics

In the course of describing an object transition, it is often highly useful to be able to provide surrounding contextual information that, while not intrinsic to the actual transition, is nonetheless closely related to it. Cataloging these context-setting objects and relations may not only be useful, but necessary. To provide this capacity, a variety of constructs from the IDEF5 ontology capture method are made available in the IDEF3 object schematic language. These constructs are entirely optional. If an analyst wishes to describe only transitions, there is no need to delve into the additional constructs discussed here. However, familiarity with these constructs provides an analyst with a good deal more expressive power. In the following subsections, the additional constructs will be presented independent of transition schematics. The integration of the two will then be demonstrated.

First-Order Schematics

Individual objects (i.e., *individuals*) are of a different logical type than the properties of those individuals. Properties are the abstract, general features that are *shared* by distinct individuals, the *respects* in virtue of which distinct individuals are the same. In a similar way, relations are the general associations which can be shared by distinct pairs (triples, etc.) of individuals. Properties and relations are identified by abstracting particular features of individuals and, hence, are often characterized as being of a *higher* (i.e., roughly, more abstract) logical type than the individuals that exemplify them. Individuals are thus frequently referred to as *first-order objects*, and properties and relations of first-order objects as *first-order properties and relations*. The *transitions-to* relation is a typical example of a first-order relation.

Displaying first-order relations between objects involves connecting two object symbols with a first-order relation symbol, as shown in Figure 3-62.

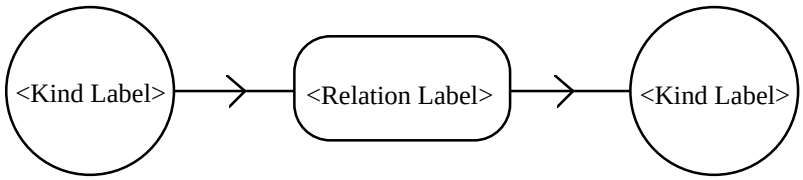

Figure 3-62
General Form of a Basic First-Order Schematic

Such schematics need a default semantics (i.e., an accepted meaning that can be assumed in the absence of any further clarification in the elaboration language). For this purpose, consider the concrete example in Figure 3-63.

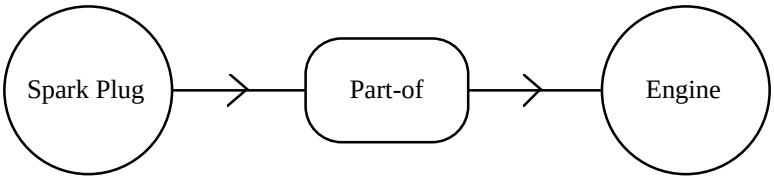

Figure 3-63
Example of a Basic First-Order Schematic

Roughly, the default meaning of this construct is a *type specification* for the *part-of* relation; that is, it specifies that spark plugs and engines are the sorts of things that can legitimately stand in that relation. It is *not* saying, for example, that every spark plug is a part of some engine, or that every engine has spark plugs; there may be loose spark plugs or plugless engines in the domain in question. Rather, in its basic, default meaning, it is simply documenting the fact that a *Spark plug* is the kind of thing that can be *Part-of* an *Engine*. If one wishes a stronger reading, it can be specified in the IDEF3 elaboration language.

As an alternative syntax for the schematics illustrated above, it is permissible (and often preferable) to replace the two connecting symbols and the relation symbol with a single arrow labeled by the same relation label, as illustrated in Figure 3-64. There is some potential for confusion here with transition schematics, but using an «open» rather than «closed and filled» arrowhead together with other particulars of the schematic should prevent ambiguity.

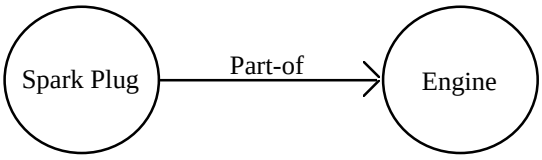

Figure 3-64
Example Illustrating Alternative Syntax for Basic First-Order Schematics

Relations like *part-of* that hold between two entities are often referred to as *2-place relations*, indicating that the number of arguments in the relation, or the «arity» of the relation, is two. However, there is no theoretical bound on the «arity» of a relation; the relation *between*, for instance, holds between three objects. More artificial but nonetheless useful relations can easily be defined with four or more arguments.

The semantics for first-order schematics involving 2-place (first-order) relation symbols generalizes to schematics involving *n*-place relation symbols. So, for example, Figure 3-65 indicates only that an instance of the *Conveys-to* relation can involve a *Conveyer*, a *Car body*, and a *Paint primer vat*.

Figure 3-65
Example of a Basic 3-Place First-Order Schematic

The numbers (optionally) attached to the spokes generalize the arrows on connecting symbols in the 2-place case. Specifically, they indicate that *Conveyer*, *Car body*, and *Paint primer vat* are to be associated with the first, second, and third argument places of the *Conveys-to* relation, respectively, as they occur in the natural English reading of the label: a *Conveyer* conveys a *Car body* to a *Paint primer vat*.

As in the 2-place case, the relation symbol can be omitted and labeled links can simply be used, as in Figure 3-66. In this document, this notation will generally be preferred.

Figure 3-66
Alternative Syntax for Figure 3-65

Though they are somewhat uncommon, relations of «arity» four and greater can be expressed in a similar fashion.

The use of individual symbols eliminates some of the indefiniteness of the schematics in Figure 3-66. For instance, the situation depicted by Figure 3-66 permits multiple paint

primer vats. However, it might be desirable in some situations to focus on one particular vat, and to represent it explicitly by an individual symbol as in Figure 3-67.

Figure 3-67
Example Illustrating the Use of an Individual Symbol

This schematic now expresses that a conveyer can convey a car body to paint primer vat PPV-1, as indicated by the individual symbol, providing a more definite proposition than the one expressed in Figure 3-66.

Indefiniteness is eliminated completely if only individual symbols are used. Thus, the schematic in Figure 3-68 is taken to express that the particular car body *CB-J27-S121* is (as opposed to only *can be*) at some time conveyed by conveyer *Conv-2* to the paint primer vat *PPV-1*.[6]

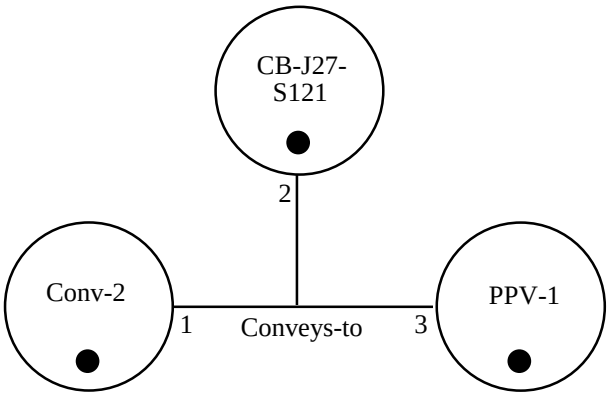

Figure 3-68
Fully Particularized Example

[6] That is, in terms of the elaboration language, Figure 3-68 translates to (conveys-to Conv-2 CB-J27-S121 PPV-1).

Multiple circles can be connected to the same circle by different arrows to create *complex* schematics. In general, complex object schematics *that do not involve transition links* are essentially just conveniences; they simply enable one to reuse graphical elements and enable one to make several assertions in the language by means of a single complex schematic. Thus, for instance, if one wished to express both that spark plugs can be parts of engines and that engines can be parts of cars, there is no need for two circles representing the kind *engine*. Rather, the two facts in question can be expressed more succinctly, as in Figure 3-69.

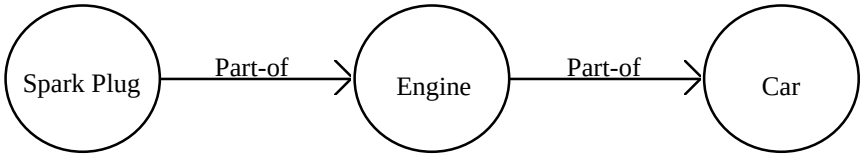

Figure 3-69
Small Complex Schematic

Similarly, one might want to add the information that, in the given domain, cars can be made in Detroit and be shipped from there to dealers. This information is conveniently expressed in Figure 3-70.

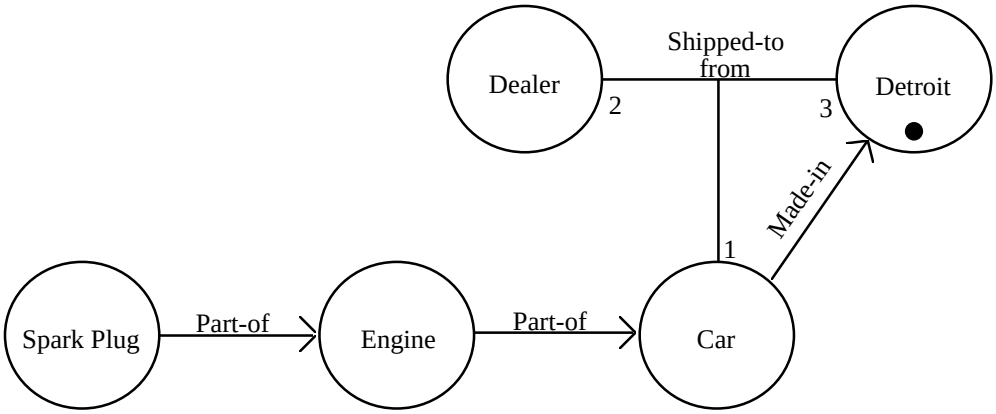

Figure 3-70
Complex Schematic Involving Multiple Relations

At the same time, an object schematic may involve only one type of relation. In such a case, to prevent needless clutter the analyst can omit labels and simply note the (single) meaning of the relation symbols at the bottom of the schematic, as illustrated in Figure 3-71.

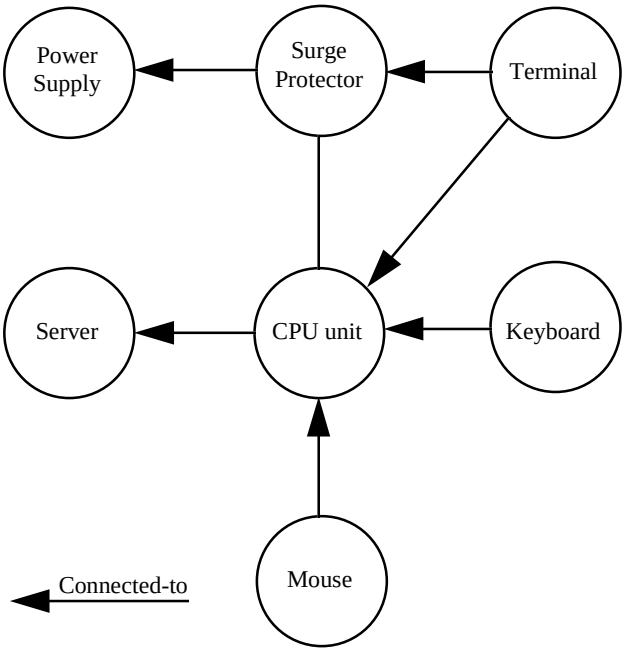

Figure 3-71
Peripheral Connections to a Personal Computer

Composition Schematics

Because the *part-of* relation is so common in design, engineering, and manufacturing ontologies, the «part-of» label and associated axioms are explicitly included in the IDEF5 languages. In particular, this capability enables users to express facts about the composition of a given kind of object. Bills of Material (BOM) are common examples of this form of expression. In general, expressing composition relations among objects is achieved by means of schematics of the form illustrated in Figure 3-72.

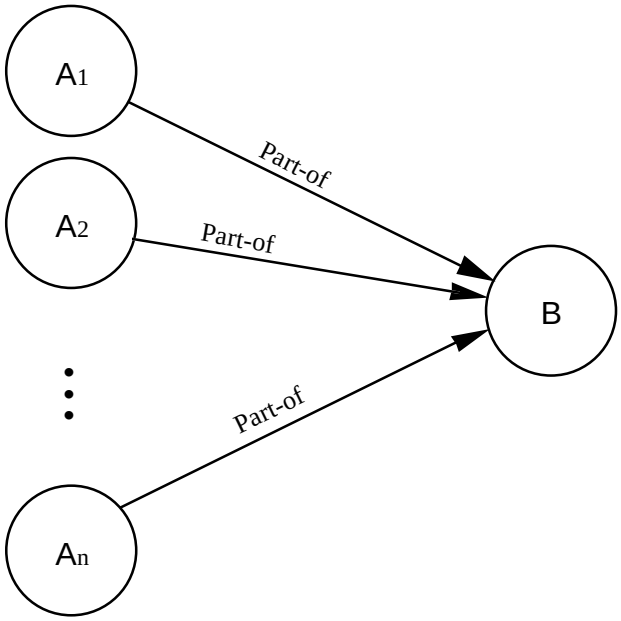

Figure 3-72
Composition Schematic

The default semantics of Figure 3-72 mean that A_1's (instances of A_1) can be parts of B's, A_2's can be parts of B's, . . ., and A_i's can be parts of B's. However, in the context of *part-of*, a stronger reading is often desired. For instance, in a BOM, one wishes to say not simply that A_1's can be parts of B's, and so on, but that every B *does* in fact consist of an A_1, an A_2, and so forth. For example, one might wish to represent the component structure for a certain kind of ballpoint pen, as in Figure 3-73.

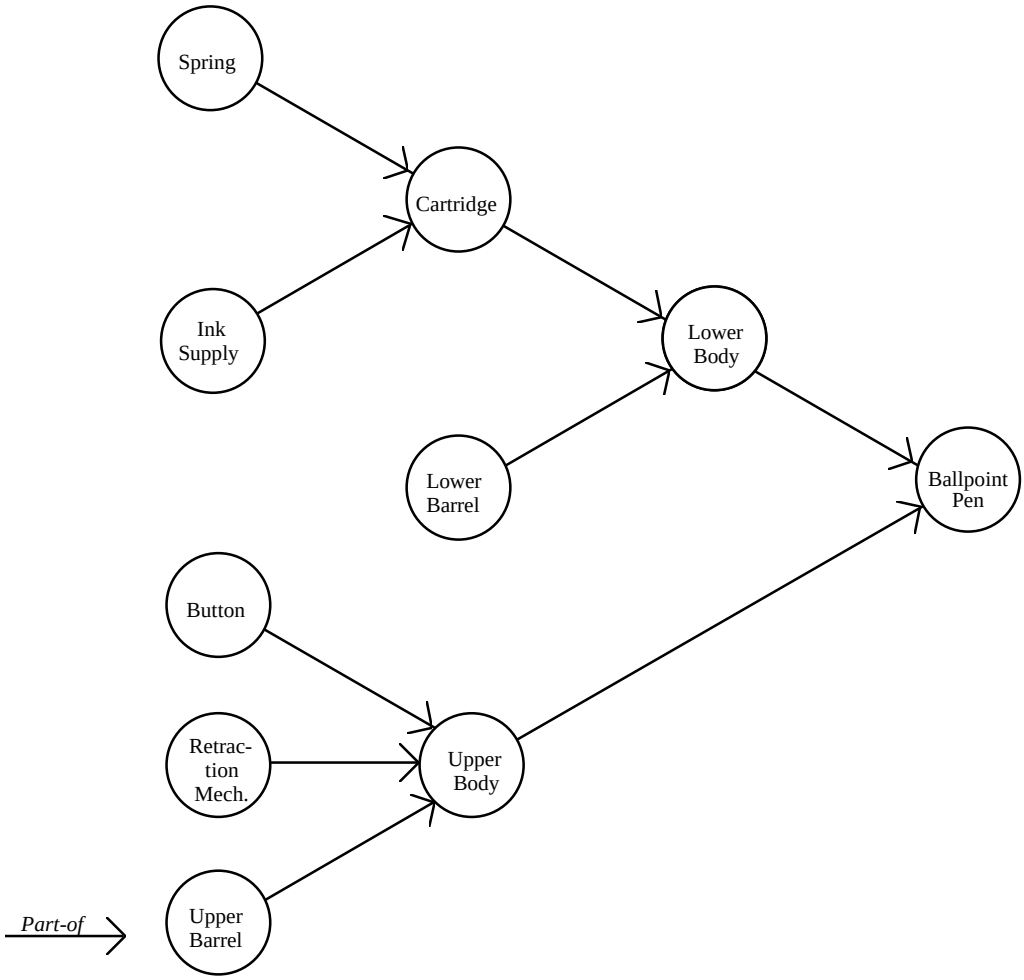

Figure 3-73
Composition Schematic

To capture this stronger meaning, one must resort to a note or to the elaboration language.[7] On this stronger semantics, then, the schematic in Figure 3-73 expresses that a ballpoint pen in the domain in question has both an upper body and a lower body, that the former consists of a button, a retraction mechanism, and an upper barrel, while the latter consists of a lower barrel and a cartridge, which in turn consists of a spring and an ink supply.[8]

---

[7] Specifically, in the case of a kind B whose instances have three parts of kinds A1, A2, and A3, one would add the elaboration language statement (forall ?x (-> (B ?x) (exists (?y1 ?y2 ?y3)(and (A1 ?y1) (A2 ?y2) (A3 ?y3) (part-of ?y1 x) (part-of ?y2 x) (part-of ?y3 x)))).

[8] Adding junctions to composition schematics also serves to narrow the range of possible interpretations. For example, using an '&' junction to 'join' multiple part-of links precludes the possibility of excluding one or more of the attached objects in the composition. In the absence of the above elaboration language statement for example, Figure 3-73 permits ballpoint pens without

Second-Order Schematics

Properties and relations that hold among individuals are identifiable (albeit abstract) objects themselves. But because they are one level of abstraction above ordinary first-order objects, they are said to be of a higher *logical type* and, hence, classified as *second-order* objects. When treated as objects, first-order properties and relations can themselves have properties. Such properties are typically known as *second-order properties*, because they apply to second-order objects. Second-order objects can also stand in relations with one another. Thus, kinds, properties, and relations that apply to individual objects are commonly known as *second-order* objects, since they are of a «higher,» more general logical order than individuals, or *first-order* objects. Like individuals, second-order objects can stand in relations to other (first- or second-order) objects. A prominent example is the *subkind-of* relation that holds between kinds, while a paradigm of a relation that holds between individuals and kinds (or properties generally) is the *instance-of* relation.

A distinct type of arrow is needed to represent second-order relations because both types of arrows connect circles, and because the associated semantics in the two cases are quite different. The basic form of a second-order schematic looks just like that of a first-order schematic, except for the presence of a so-called *second-order relation arrow* (as shown in Figure 3-74) instead of a first-order relation arrow.

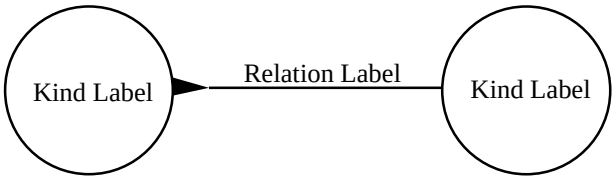

Figure 3-74
Basic Second-Order Schematic

The semantics for second-order schematics is much more definite than the semantics for most first-order schematics. Specifically, second-order schematics are about the indicated kinds, rather than about their instances: in Figure 3-74 the kind represented by the left-hand circle stands in the (second-order) relation indicated by the arrow with the kind represented by the right-hand circle. Furthermore, the default semantics are not qualified; unlike general first-order schematics, the semantics are not merely about how things can be in the domain but about how two kinds are in fact related.

---

springs and retraction mechanisms. By adding junctions to the schematic, the analyst can indicate that, for example, springs and ink supplies can be parts of cartridges for ballpoint pens but cartridges without springs cannot exist, and so forth.

The schematic in Figure 3-75 expresses that there are more U.S. citizens than Canadian citizens (i.e., more literally, that the kind *U.S. Citizen* has more instances than the kind *Canadian Citizen*).

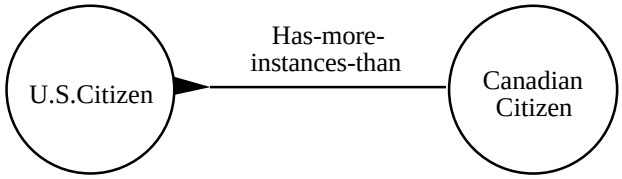

Figure 3-75
Example of a General Second-Order Schematic

Figure 3-76 illustrates a schematic involving the second-order relation *subkind-of*. By the semantics just given, the kind *hex-headed bolt* is a subkind of the kind *fastener*.[9]

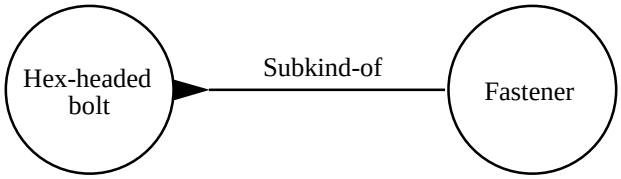

Figure 3-76
Example of a Second-Order Schematic with *Subkind-of*

Classification Schematics

Because the subkind relation is so common, the default meaning of the second-order relation arrow with no associated label represents the subkind relation, thus permitting users to avoid having to attach the label *subkind-of* repeatedly throughout a schematic. This choice is motivated by the observation that among the more common mechanisms for representing knowledge are taxonomy diagrams (Brachman, 1985). Domain experts engaged in knowledge acquisition often make statements such as *A is a B, A is a type of B,* or *A is a kind of B*. The cognitive activity involved in organizing knowledge in this fashion is called *classification*. There are several identifiable varieties of classification. Two particularly prominent types of classification are *description subsumption* and *natural kind classification*. In description subsumption, (1) the defining properties of the «top-level» kind *K* in the classification, as well as those of all its subkinds, constitute rigorous necessary and sufficient conditions for membership in those kinds, and (2) the defining properties of all the subkinds are «subsumed» by the defining properties of *K* in the sense that the defining properties of each kind entail the defining properties of *K*; the defining properties of *K* constitute a more general concept.

[9] In terms of the elaboration language again, we have simply (subkind-of hex-headed-bolt fastener).

In natural kind classification, by contrast, it is not assumed that there are rigorously identifiable necessary and sufficient conditions for membership in the top-level kind *K*, but that, nonetheless, there are some underlying structural properties of its instances that, when specialized in various ways, yield the subkinds of K. The best examples of such classification schemes are, of course, genuine natural kinds such as *metal*, *feline*, and so forth, but the idea can be extended to artifactual kinds like *automobile* and *NC machine*. These two types of classification are illustrated in Figure 3-77.

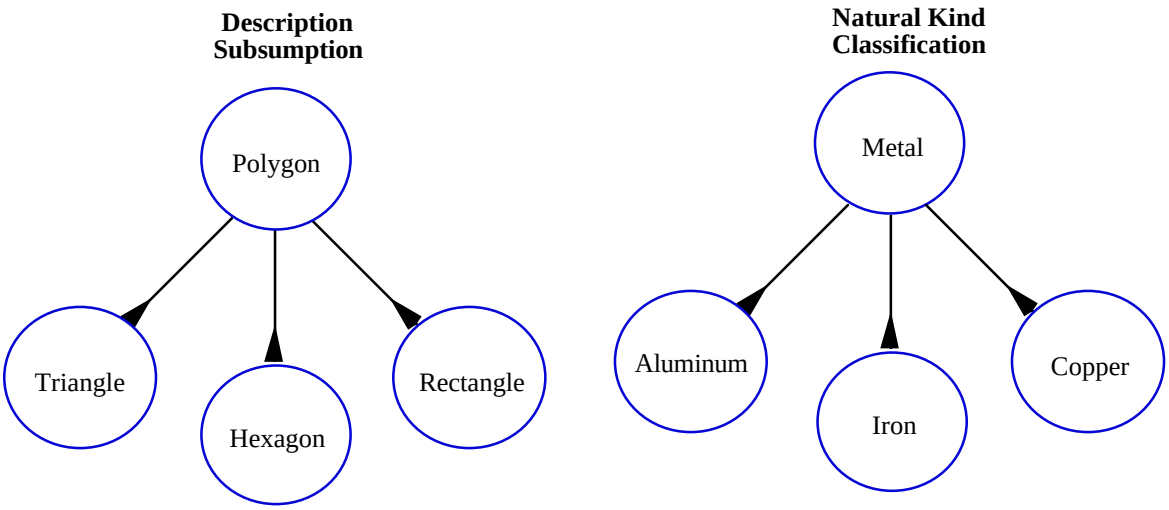

Figure 3-77
Different Types of Classification

Clearly, with its central notion of a kind, a natural application for the general object schematic language is the development of taxonomy diagrams, or as we shall call them, *classification schematics*.

Classification is typically much more detailed than the examples suggest. Most classification schemes will involve several levels of more specialized subkinds «below» more general kinds in the scheme. (Both the *subkind-of* and *instance-of* relations are often ambiguously expressed by the relation «is-a» in semantic nets and other graphical languages. Such schemes are often called 'is-a hierarchies,' but the use of 'is-a' is strongly discouraged; either the *subkind-of* relation or the *instance-of* relation should be used instead, depending on the intended meaning.) To illustrate, it is essential in project planning that one categorize the kinds of *resources* that will be needed for the project's success. Informally, a resource can be defined as an object that is consumed, used, or required to perform activities. Resources play an enabling role in processes. Classification schematics provide a natural way of categorizing necessary resources, as, for example, in Figure 3-78.

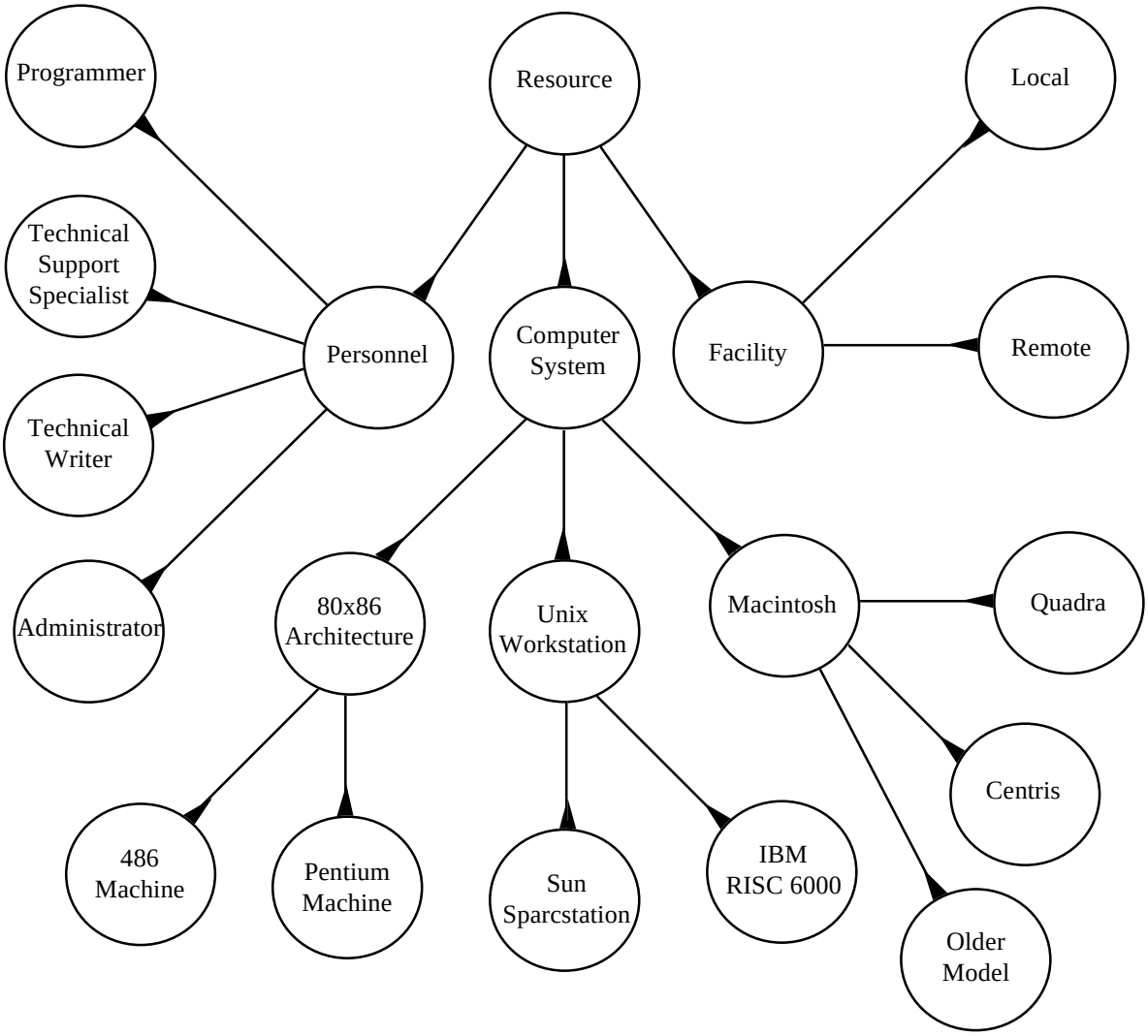

Figure 3-78
Classification of Resources

Hiding Composition and Classification Information

As illustrated above, use of composition relations can yield quite detailed schematics. Such detail can cause a great deal of clutter. For instance, in addition to describing the component structure of the kind *ballpoint pen*, one might also want to talk about many of the other relations it and its instances are involved in (for example, that the pens can be made in Sequim, Washington, that fountain pens generally cost more than ballpoint pens, that *ballpoint pen* is a subkind of *pen*, and so on). In many contexts, the component structure of the kind might well be irrelevant, and in such cases it would be useful to be able to hide that information. That such information is being hidden is indicated on a diagram by using a double circle (instead of a standard single circle) to represent the kind,

along with a ‘P’ (for *part-of*) in the top of the circle to distinguish the kind of information that is being hidden, as illustrated in Figure 3-79.

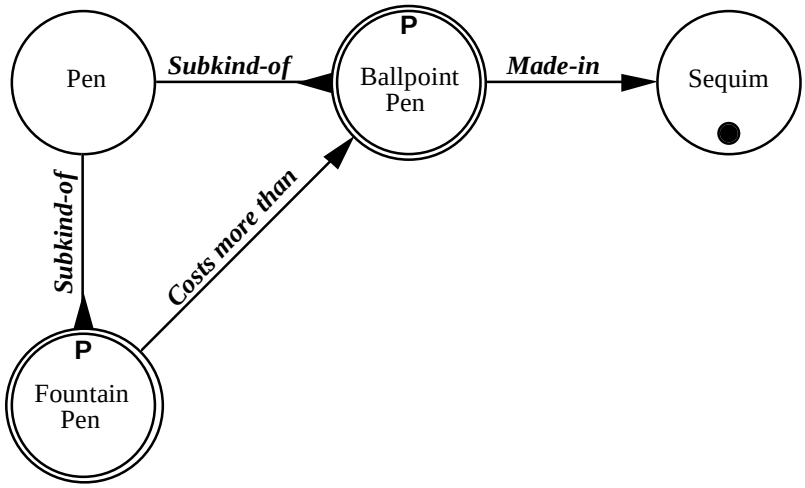

Figure 3-79
Hiding Composition Information

This example illustrates the use of first- and second-order relation symbols in the same schematic.

In a similar fashion, it often proves useful to hide classification details in an object schematic. In some contexts (e.g., those in which facilities and personnel need to be highlighted), information about computer systems might not need to be explicit. As with the composition relation, hidden information will be indicated by a double circle, annotated in this case with a ‘C’ (for ‘classification’) at the top of the circle. Thus, one might alter Figure 3-78 by hiding information about computer system subkinds and adding information about facilities to obtain the schematic illustrated in Figure 3-80.

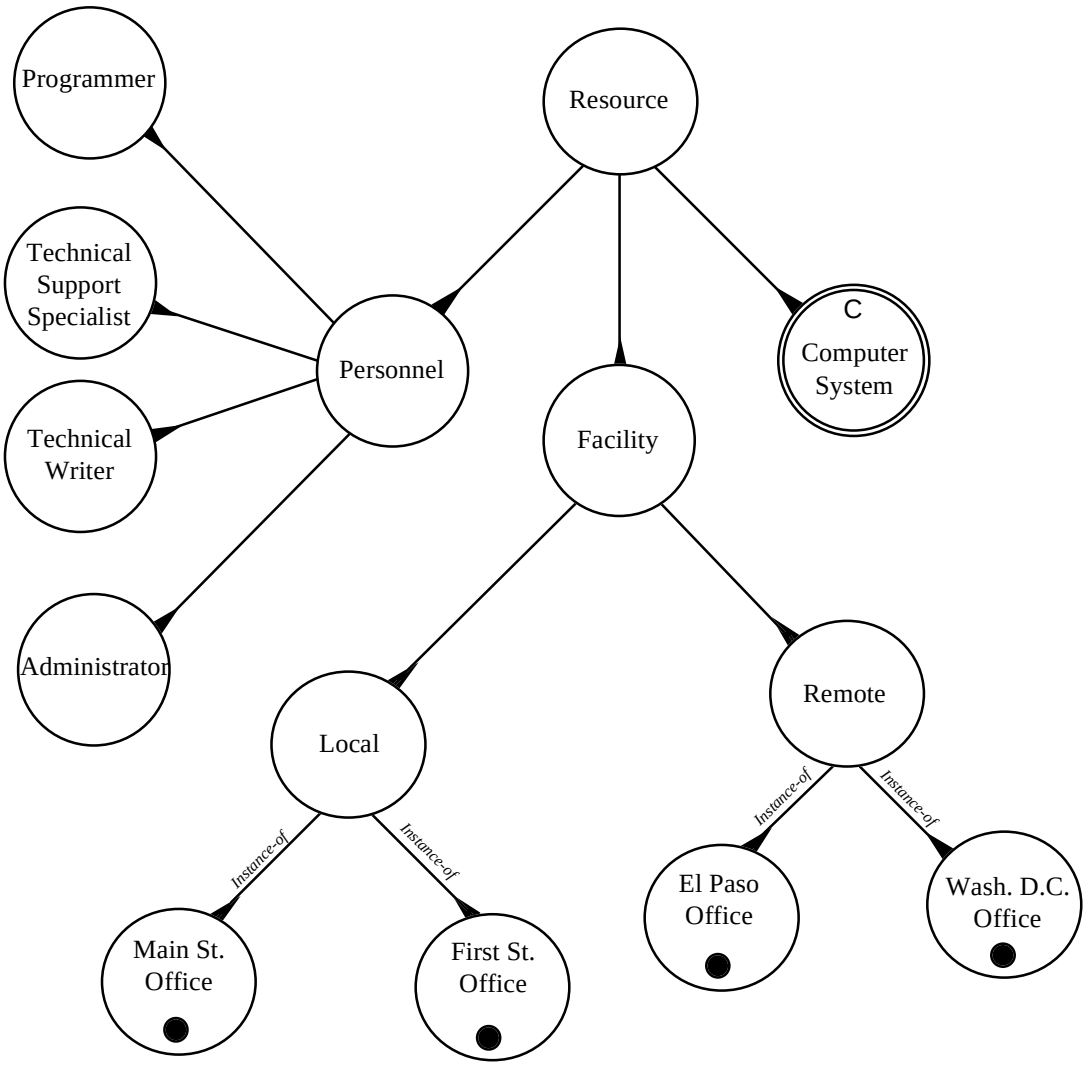

Figure 3-80
Classification of Resources with Hidden Information

Creating Enhanced Transition Schematics

The general schematic constructs can be applied to enhance transition schematics with additional information useful to the context and purpose of the description development effort. Using the Transition Schematic as the central focus, context-setting information is added as illustrated in Figure 3-81. Here, the states through which water traverses in a heating process are represented. The *subkind-of* relation has been added to the schematic, illustrating that the kind *water* has subkinds represented by the various object states.

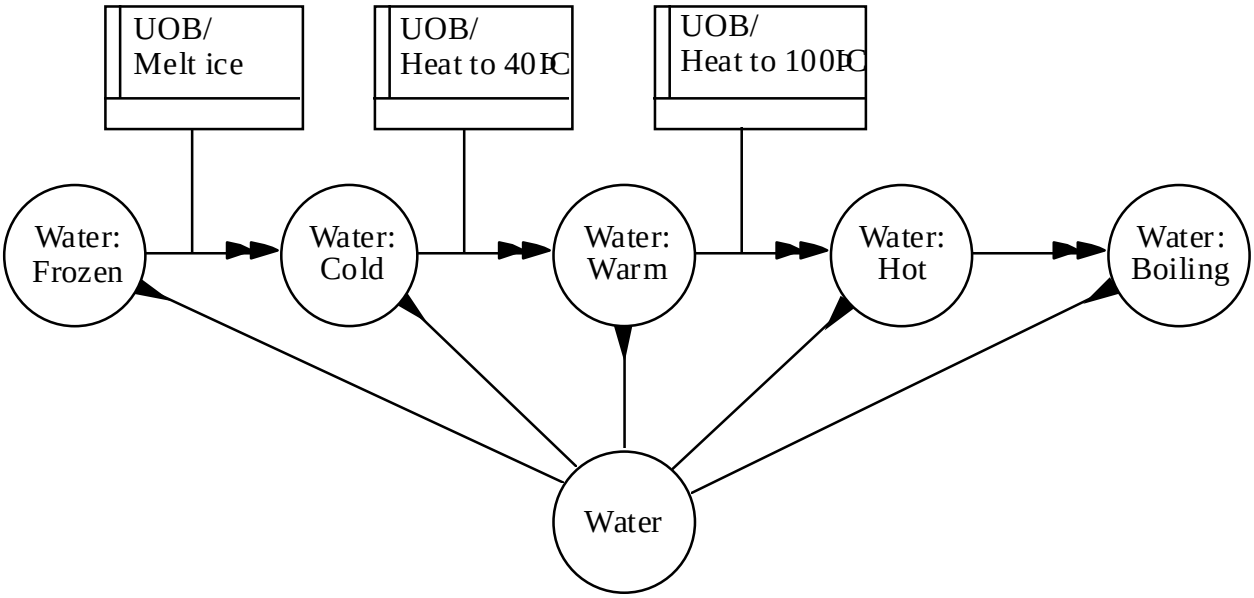

Figure 3-81
Combined Schematic Displaying States and Transitions

Another simple example of a schematic that integrates general object schematic constructs with transition schematics is seen in Figure 3-82.

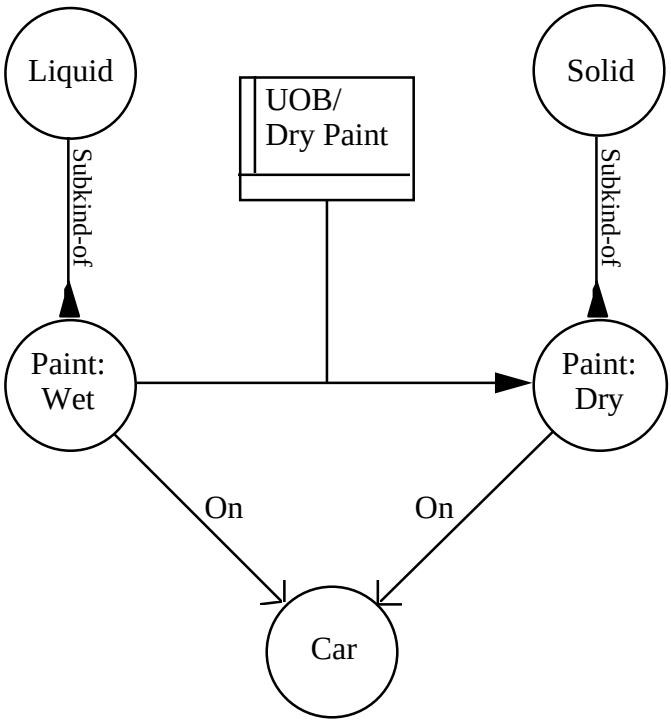

Figure 3-82
Object Schematic Involving Object Transition Constructs

In this example, in addition to indicating the transition of a quantity of paint from a wet to a dry state via a drying process, relation symbols are used to indicate that the states *Paint:Wet* and *Paint:Dry* are also related to other kinds, as indicated by the labels on the arrows.

For a more complex example, consider the schematic shown in Figure 3-83.

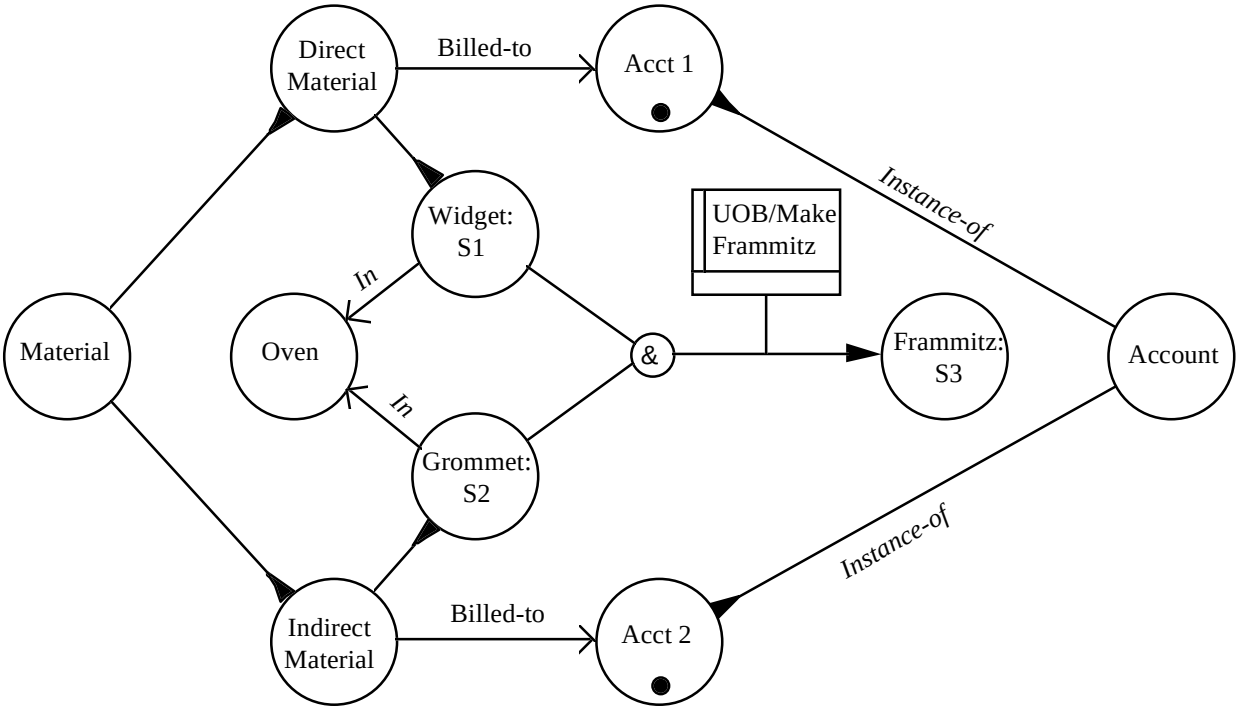

Figure 3-83
Another Object Schematic Involving Object Transition Constructs

At its center, the schematic represents a transition in which a *Widget* in state *S1* and a *Grommet* in state *S2* participate in a *Make Frammitz* process that yields a *Frammitz* in state *S3*. The use of relation links, however, enables one to express in addition that the widget and grommet are in an *Oven* in the process. Also, one is able to express a good deal of additional contextual information, such as (1) widgets are *Direct material* whereas grommets are *Indirect material*, (2) both of those are kinds of *Material*, (3) to what accounts such materials are billed, and so on. The general object schematic constructs introduced above enable an analyst to fill in a wide variety of contextual details surrounding a given transition.

It is important to observe that the above interpretation of the schematic in Figure 3-83 is not the only possible interpretation. Most notably, how is one to determine when a piece of information added to a transition schematic holds only relative to the transition in question, and when it holds in general. For instance, the *In* link between *Widget:S1* and *Oven* was interpreted to mean that *in (instances of) the indicated transition*, the widget in the transition is in an oven. There is nothing in the schematic proper that prevents one from interpreting this to mean that, at all times, a widget is in state S1 when and only when it is in an oven. Similarly, there is nothing about the schematic that determines whether widgets are always considered direct material, or only that the widgets used in instances of the transition in question are so considered. Widgets in other contexts (in the same state) might be considered indirect material. In IDEF3, the general interpretation

will be taken as a *default*. That is, unless otherwise noted, the relations recorded in a schematic will be taken to hold in the widest possible context. If a narrower context is intended, this can be recorded in a note, or more formally in the elaboration language.

Elaborations

The elaborations that can be attached to the schematic elements (e.g., UOB boxes, junctions, links, object symbols) are critical to understanding a process description. Elaborations provide detailed characterizations of the entities referred to by the schematic element in question. These detailed characterizations are presented in an *elaboration document.* Elaboration documents typically include: (1) the schematic element's name, label, and number; (2) listings of the *object types and instances*, *facts*, and *constraints* that are associated with the entity the element signifies; and (3) a textual description of that entity.

The distinction between facts and constraints deserves some clarification. A fact is simply a statement that has been observed to hold in at least one instance of a process. For instance, in a paint/dry process, an instance of the *paint-part* UOB might have been observed to have a duration of 4.5 minutes; or, the color of a part entering the process might have been observed to be gray. Descriptive facts like these simply record what has happened in some instances of the UOB in question.

A large store of descriptive facts is useful in the early stages of building an IDEF3 description. These facts generally serve as the raw data from which a more definite and accurate process description emerges. As knowledge of a process grows, facts are needed to record not only what has happened in some instances of the process, but what *must* happen in *all* instances of the process. These facts are called *constraints* in IDEF3. Because a fact that *must* hold also holds as a matter of contingent fact, constraints are thus a special kind of fact. Two broad sorts of constraints can be distinguished: *absolute* and *conditional*. An absolute constraint is stated without qualification (e.g., *All pieces of mail must have a zipcode displayed*). Conditional constraints are conditional in form: IF a certain state of affairs A holds, THEN a certain other state of affairs B must hold as well. For example, it might be a constraint in a paint/dry process, that IF (in an instance of the paint-part UOB) the object being painted is of kind K, THEN the duration of the paint-part instance must be exactly 5 minutes. An object of another kind, by contrast, might be painted for only 4 minutes.

Every identified element in a process description has an elaboration document associated with it. The elaboration document may consist of only a reference number and, optionally, a label. However, by adding more information, the elaboration document provides an important key to understanding the elements that constitute complex

processes. A detailed discussion of elaboration documents and their contents is provided in Section 4, Developing IDEF3 Descriptions.

Generally speaking, elaboration documents are populated with natural language statements. When something more structured and precise than natural language statements is required in the elaboration, users can use IDEF3's elaboration language. The elaboration language will be illustrated with two examples in the following section. See Appendix A for a complete account of the elaboration language and further examples.

Some Examples of the Elaboration Language

The elaboration language is a logical language based (with some modifications) on a subset of the emerging information-sharing standard known as the Knowledge Interchange Format (KIF) (Genesereth & Fikes, 1992). This subset is known as the elaboration language *core* which contains the basic elements needed for almost any logical language. The core is extended by a number of IDEF3-specific constructs designed to express precise information about the processes and transitions represented in the IDEF3 schematic languages. However, for this to be done effectively, it is essential to have a clear *semantics* for the language. The intuitive semantics for IDEF3 schematics are based upon *situation theory*, a recently developed theory of information [(Barwise & Perry, 1983); see Section A.4 of Appendix A for an informal overview of the theory]. In the elaboration language, basic concepts of IDEF3 such as UOB, process, and the like are identified with certain basic semantic categories of situation theory. The constructs added to the elaboration language core correspond to these categories.

To illustrate the use of the elaboration language in conjunction with a process schematic, consider the process in the schematic in Figure 3-84. Call this process «PQD».

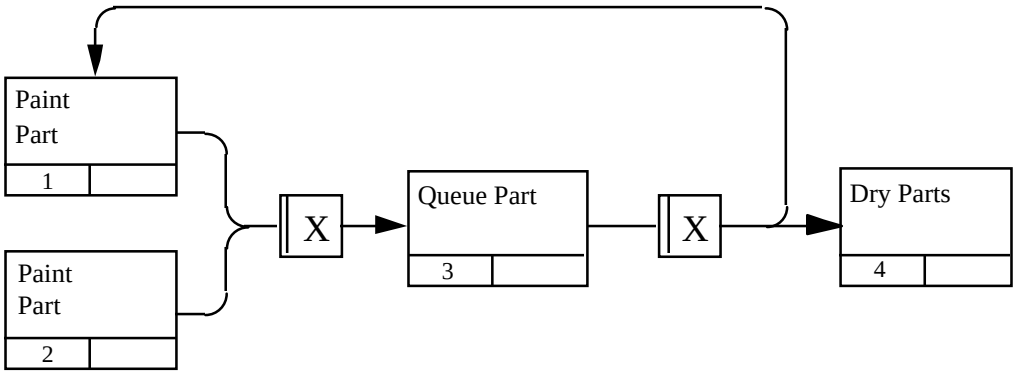

Figure 3-84
Paint/Queue/Dry Process

In addition to constraints indicated in the schematic, there could be a wide variety of additional constraints on the process that cannot be expressed in the graphical language, for instance:

In an activation of PQD, exactly one part is painted in any given occurrence of ***Paint Part****.*

This constraint is expressed as follows:

```
(forall      (?coe : (activation-of ?coe PQD))
   (forall   (?sit : (and (occurs-in ?sit ?coe) (occurrence-of ?sit Paint-part)))
             (exists!-1 ?x (supports ?sit (painted ?x +)))))
```

In this constraint, the variable «?coe» ranges over *courses-of-events*, i.e., activations, or instantiations, of general processes like PQD. The variable is further restricted by the expression to the right of the colon—(activation-of ?coe PQD)—to those courses of events that are activations of the process PQD. Then for any such course of events c, the remaining two lines then say that, for any situation s that is an occurrence of the UOB, or situation type, *Paint Part* in c, there is exactly one object (the meaning of «exists!-1») x such that x is being painted in s—i.e., in the language of situation theory, such that s *supports* the information that x is being painted.

Note that this constraint, as expressed, applies to the entire PQD scenario depicted in the diagram. However, it may be more natural to add the constraint directly to the characterization of *Paint Part*, where it is intended to apply to each occurrence of *Paint Part* in a given activation of PQD. The general universally-quantified conditions at the beginning of the constraint can thus be dropped and the constraint can be expressed much more simply and directly as follows:

```
(exists!-1 ?x (supports ?sit (painted ?x +))).
```

Note that, as a constraint on *Paint Part*, the situation variable «?sit» is not thought of as implicitly universally-quantified but rather as a parameter playing the role of a given occurrence of *Paint Part* in a given activation; similarly for the object variable «?x».

A second example presupposes that a number of auxiliary notions have been defined, viz., the relation *in-queue* — which holds between an object, a queue, and an interval just in case the object is in the queue during the interval — and a function *start-of* that takes a situation to the point (a variety of interval) in time at which it starts. The elaboration language provides powerful facilities for creating such definitions. Consider, then, the following constraint on PQD.

In an activation of PQD, no instance of ***Paint Part*** *begins at any time if there are five objects in the queue at that time*.

```
(forall      (?coe : (activation-of ?coe PQD))
    (forall  (?sit : (and (occurs-in ?sit ?coe) (occurrence-of ?sit Paint-part)))
             (not (exists-5 ?x (and  (instance-of ?x Part)
                                     (supports ?sit (in-queue ?x Q (start-of ?sit) +)))))).
```

That is, for any activation of PQD there is in that activation no occurrence s of *Paint part* such that there are five (or more) objects in the queue at the start of s. Again, this constraint is expressed generally about PQD, but if it is added directly to the characterization of *Paint Part* where it is intended to apply to the occurrences of *Paint Part* within a given activation, it can be expressed directly as follows:

```
(not (exists-5 ?x (and (instance-of ?x Part) (in ?x Q (start-of (interval-of ?sit)))))).
```

To illustrate the use of the elaboration language with object schematics, consider the enhanced transition schematic in Figure 3-83. As noted above, in such schematics there is some semantic indeterminacy as to the scope of the surrounding contextual information. For example, are grommets generally considered indirect materials or are they so considered only in more restricted contexts like the depicted process? Such information can be added explicitly in the elaboration language. Thus, the following constraint might be added explicitly to the elaboration document for the schematic, expressed as indicated.

Grommets are considered Indirect Materials in all situations.

```
(forall      (?sit ?x : (supports ?sit (grommet ?x +)))
    (supports ?sit (indirect_material ?x)))
```

That is, any situation at all (relative to the given enterprise) that supports the information that x is a grommet also supports the information that it is indirect material.

As noted previously, an object symbol in a transition schematic indicates, in addition to the state in question, the type of situation in which an object is in that state. Hence, enhanced transition schematics are a bit ambiguous with regard to the meaning of object symbols. For example, in Figure 3-83, in the context of the embedded transition schematic, the *Widget* symbol indicates the type of situation in which there is a widget, whereas, in the context of that symbol being linked to the *Direct Material* symbol, widgets are indicated as a kind of direct material. To sort out this ambiguity in the elaboration language, we will use the term «Widget*» to signify the type of situation in which there is a widget.

Given this, it is now possible to illustrate how one would use the elaboration language to express the following constraint.

The widget and the grommet in an instance of WGF are in the oven at 500 degrees for a period of 5 minutes before they are assembled into a frammitz.

```
(forall      (?coe : (activation-of ?coe WGF))
                        (forall (?sit ?sit1 ?sit2:       (occurs-in ?sit ?coe)
                                (occurs-in ?sit1 ?coe)
                                (occurs-in ?sit2 ?coe)
                                (occurrence-of ?sit Frammitz*)
                                (occurrence-of ?sit1 Widget*)
                                (occurrence-of ?sit1 Grommet*))
            (forall  (?x ?y) : (supports ?sit1 (Widget ?x))
                               (supports ?sit2 (Grommet ?y)))
                (exists (?sit3 ?oven: (during ?sit1 ?sit3)
                                      (during ?sit2 ?sit3)
                                      (precedes ?sit3 ?sit)
                                      (supports ?sit3 (Oven ?oven)
                                      (supports ?sit3 (= (temp-of ?oven) 500)))
                        (and (supports ?sit3 (in ?x ?oven))
                             (supports ?sit3 (in ?y ?oven))
                             (supports ?sit3 (= (in-oven-during ?x 5)))))))
```

That is, in any occurrence c of WGF, if ?sit, ?sit1, and ?sit2 are occurrences of Frammitz*, Widget*, and Grommet*, respectively, in c, then if ?x and ?y are the Widget and the Grommet in Widget* and Grommet*, respectively, then there is an object ?oven and a situation ?sit3 such that (1) ?sit1 and ?sit2 occur during ?sit3, (2) ?sit3 precedes ?sit, and (3) ?oven is an Oven whose temperature is 500 degrees in ?sit3, and such that ?x and ?y are in the oven in ?sit3.

Notes

A note box may be attached to a UOB, junction, object, link, or referent. Notes allow the IDEF3 analyst to perform the following.

1. Emphasize the participation of particular objects or relations associated with the attached UOB or junction.

2. Tie in specific examples of referenced data or objects (e.g., screen layouts).

3. Highlight special constraint sets associated with a given junction elaboration. Notes can be used to call attention to, or list the

contents of, a junction elaboration (e.g., additional facts, constraints, or decision logic which describe how that junction works).

Notes may be used to provide additional information about a particular IDEF3 model element or to attach illustrations, text, screen layouts, comments, etc. to the description. New IDEF3 users will often find that notes provide an easy way to express ideas or concepts in lieu of junction types, dashed arrows, or constraint language statements.

The example in Figure 3-85 illustrates how a note can be used to highlight the association of special constraint sets with junctions. This description states that, for certain conditions, it will be required to loop back to UOB *Perform Mission Area Analysis*. In this case, the note on Junction J1 is used to display the conditions under which the referent *UOB/Perform Mission Area Analysis* would be activated.

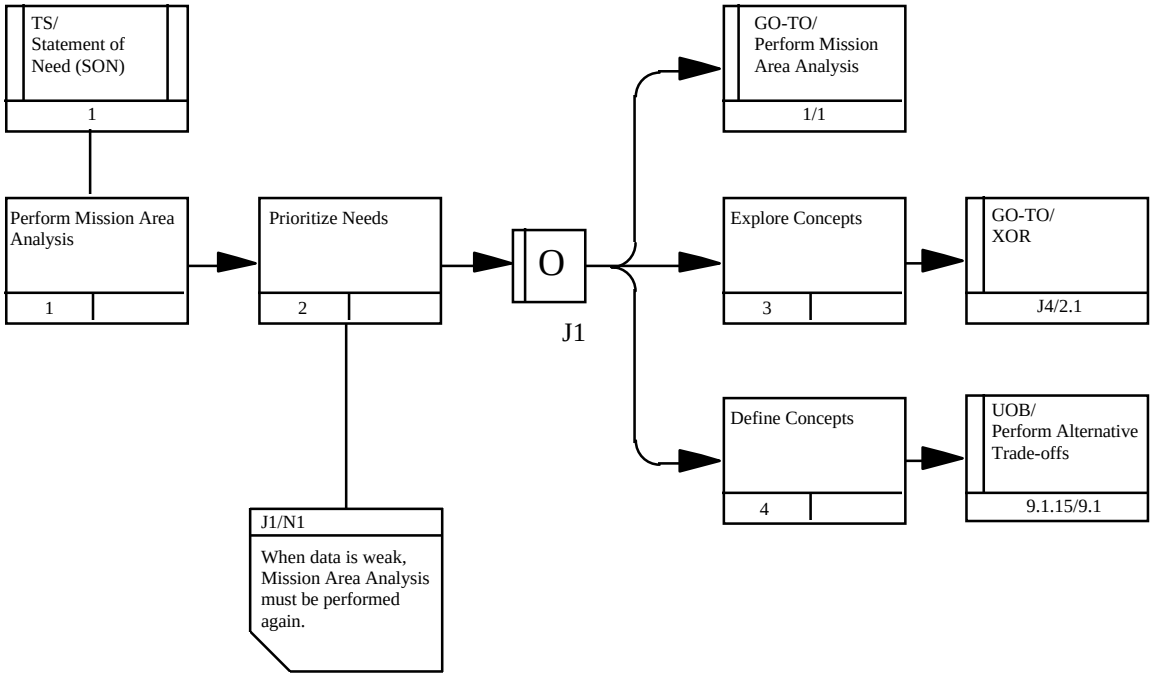

Figure 3-85
Note Associated with a Junction

The note box is divided into two sections. The band across the top of the note is used for note identification. It contains a Note ID comprised of the referenced element number and the Note number (e.g., J1/N1). The bottom section of the note box, called the note field, is provided for the note itself. No formal structure is imposed on the contents of this field, although authoring conventions established for project-specific purposes are permitted. For example, the note field may be structured to provide references to a set of notes associated with the referenced element, thereby permitting a restriction of no more than one note for each schematic element. Alternatively, the note field may be structured

to begin with some kind of note classification, thereby alerting readers to the main focus of the note.

Representing Stochastic Processes

Markov chains and other types of stochastic processes are obvious candidates for representation via object schematics with logical branches. Such processes consist of a set of states of a given system S together with, for any state A1, and for any other (possibly the same) state A2, the probability that S will transition from the former to the latter. A system is a certain kind of complex object; hence, it is natural to represent the possible transitions from A1 to any other state as a sort of exclusive disjunction as in Figure 3-86, where, in addition, real numbers representing probabilities have been assigned to each transition link extending from the XOR junction.

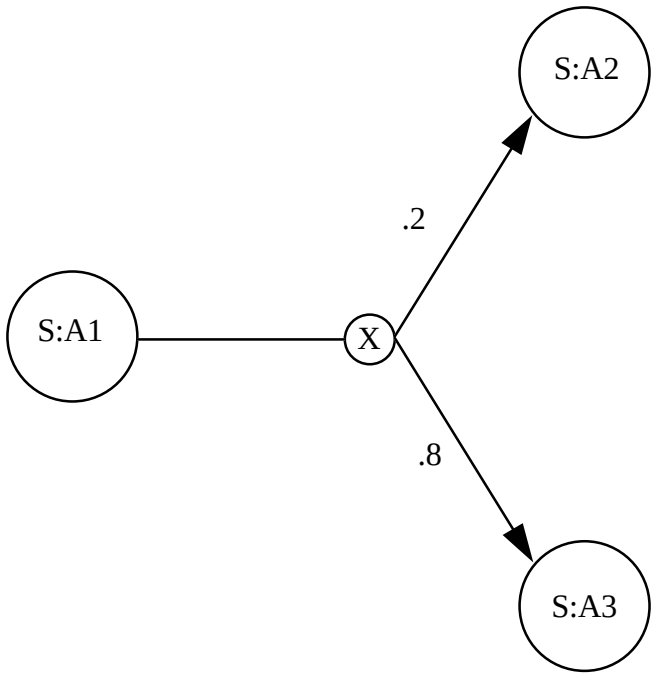

Figure 3-86
Transition Schematic Illustrating Possible Complex State Transition Logic

There is such a schematic for each state A_n. Although this is illustrative in each case, it is clearly more efficient, and less cluttered, to represent the entire process—the entire collection of probabilistic transitions from any state to any other—in terms of the more standard graphical representation in which arcs extend directly between any two given states in both directions.

There is no reason why Transition Schematic syntax cannot be adapted to permit such representation *as a convention,* that is, as a sort of abbreviation for the collection of all Transition Schematics like the one in Figure 3-86. Use of IDEF3 transition links enables one to augment such schematics with actual descriptions of the processes that effect the transitions in question. The convention in question is illustrated in Figure 3-87; probabilities are suppressed to reduce clutter.

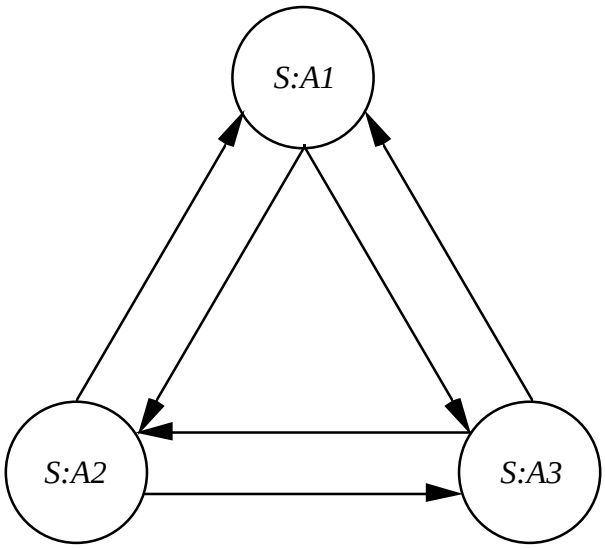

Figure 3-87
Transition Schematic Convention for Representing Stochastic Processes

DEVELOPING IDEF3 DESCRIPTIONS

This section presents a procedure for using IDEF3 as a process description capture, consolidation, and validation method. The procedure is targeted at the needs of a large effort involving a team approach; projects more narrow in scope may not require all of the activities described here. Because the application procedure depends largely on the purpose for which the method is being used, project leaders are encouraged to prepare a detailed method application guide at the beginning of the project.

The description development procedure is presented first in terms of the evolutionary cycle through which IDEF3 descriptions are realized and then in terms of a functional description amenable to project phasing.

The IDEF3 Description Evolution Cycle

Developing IDEF3 descriptions involves the creation of Process Schematics, Object Schematics, and their associated elaborations. The description capture and validation process is highly recursive and iterative. As with any recursive process, process termination criteria are important—i.e., it is important to know when to stop. Although it is not possible to give precise criteria for the completion of description development activities, some basic guidelines apply. First and foremost, description development is generally undertaken to accomplish some purpose. That purpose may be simply to document a process—in which case the development of schematics and elaborations is an end unto itself. In most cases, however, description development is undertaken to assist with some discovery or decision-making activities. In these situations, the time and effort spent on description development will be determined by the information needs of the project. Whether IDEF3 is used to document a process or to assist with discovery and decision-making, descriptions approaching completion exhibit increasingly reduced rates of change in terms of their structure, scope, and level of detail.

The development of IDEF3 descriptions is a process of capturing knowledge about how activities are performed in a given organization. In general, when using IDEF3 to collect and organize these descriptions, the following five steps are applied recursively.

1. **Collect:** Acquire observations and written descriptions of both process instantiations and generalizations across process instantiations.

2. **Classify:** Individuate situation types, objects, object types, object states, and relations.

3. **Organize:** Assemble the data that has been collected and classified using IDEF3 structures.

4. **Validate:** Ensure that the statements made in IDEF3 are grammatically correct and that they corroborate with the collected descriptions of the actual or idealized situation.

5. **Refine:** Make adjustments to the existing structures to incorporate newly discovered information, to simplify the presentation, or to highlight important elements of interest.

Recursive application implies that the same development process continues until the information and knowledge available in the domain has been collected and organized into a structure that satisfies the termination conditions of description development.

IDEF3 Description Capture Activities

Experience with IDEF3 indicates that description capture is similar to knowledge acquisition and design endeavors. It is highly iterative, driven by findings, and often stylized by the participants. The activities described in this section should be considered «modes of thought» rather than sequential steps. The user should not expect to apply these activities in a strictly sequential manner. With these ideas in mind, the framework presented in this section provides a default structure for first-time IDEF3 users.

Define the Project

The development team must establish the purpose and context of the description capture effort as early as possible in the project. The purpose statement provides a *completion criteria* for the description capture effort. The purpose is usually established by a list of (1) statements of objectives for the effort, (2) statements of needs that the description must satisfy, and (3) questions or findings the client wants answered. The context statement bounds or delimits the area of the domain addressed by the project. The context is established by scope statements and the identification of the initial scenarios for the description capture project.

The purpose and context can rarely be determined completely in advance. The client often revises his list of needed findings or questions as data compilation begins. The area an analyst thinks will lead to the answer often turns up leads in other areas that were considered out of scope. The purpose and context generally evolve during the initial part of the project. The purpose and context of an IDEF3 description are captured on an IDEF3 Description Summary Form similar to the one shown in Figure 4-1.

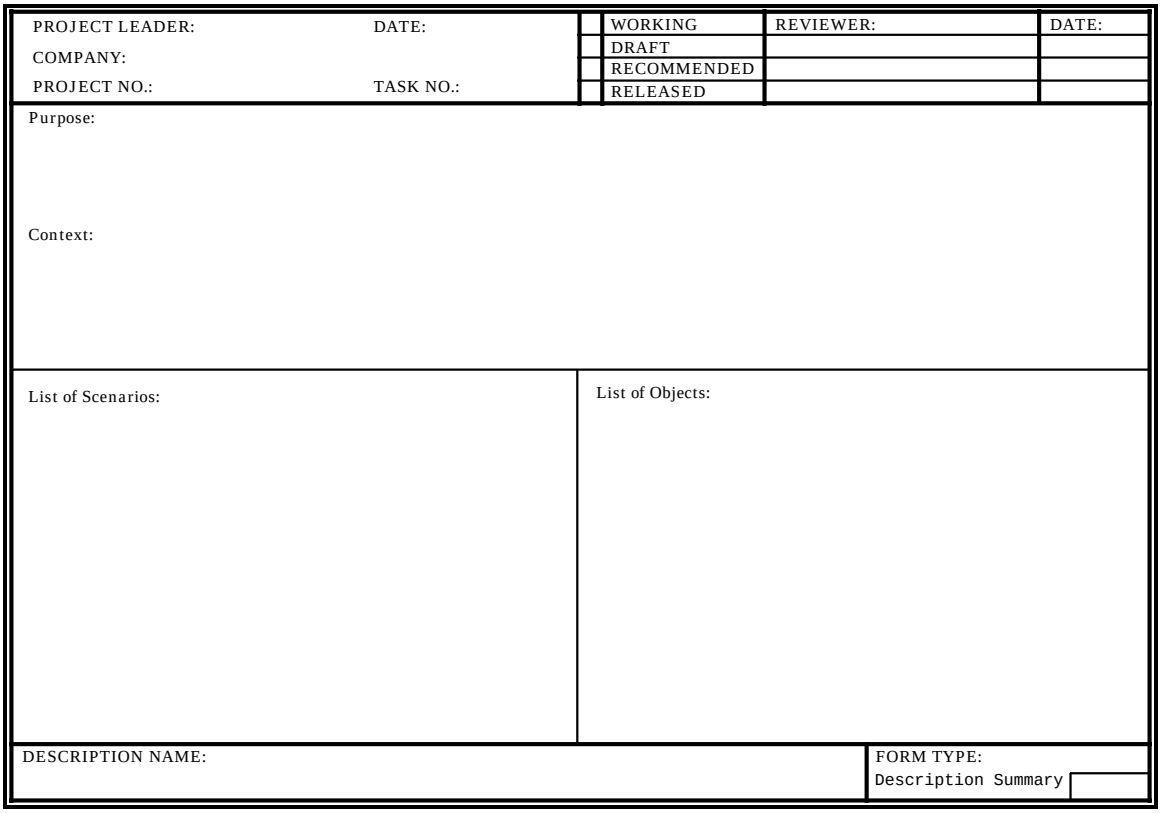

Figure 4-1
IDEF3 Description Summary Form

Define the Purpose

Defining the purpose is an important initial step in the development effort. Without a purpose statement, the only completion criteria is the budget and time allocated to the effort. Defining the purpose can be separated into two parts: (1) defining a Needs Statement and (2) defining the information goals in terms of how that descriptive information will be used.

The Needs Statement should identify the source of the request (person or project) and paraphrase the stated objectives of the client. Identifying the information goals is simplified by answering the following questions:

1. Who will use the description once it is available?

2. What question(s) does the client need answered?

3. What issues are behind the need for the process description?

4. What decisions are behind the need for the process description?

Establish the Context

Once the purpose of the effort has been characterized, it is possible to define the context of the project in terms of the scope of coverage and level of detail.

Defining the context of the project begins with defining the boundaries of the description capture effort and documenting those boundaries in a set of scope statements. Specifying project scope involves defining which parts of the system are to be included and which are to be excluded. Ideally, the scope should identify only those areas relevant to the needs of the client.

An effective mechanism for defining the scope of the project is identifying the important scenarios of operation to be considered and those that, although related, fall outside the project boundaries. Identifying a scenario involves achieving a consensus among the team members on a title and paragraph description of a commonly-occurring situation or problem that the system (organization) addresses. It is common for different scenarios to represent alternative viewpoints of essentially the same process. When possible, the beginning and ending UOBs of the scenarios should be established. Additionally, activities that impact or feed the scenarios, but are outside the context of the description, should be identified to further refine the boundary of the description capture effort. While the statements of purpose and scope provide useful guidelines for the successful completion of this activity, the insight of domain experts must be relied upon to actually identify the scenarios. The project leader should be aware that the scenarios identified are still at a tentative level and that some change can be expected as the data is collected and analyzed.

An activity closely related to defining the scope is determining the level of detail of the description capture effort. The required level of detail is determined by identifying what detail is needed to resolve an issue, make a decision, or answer a question. The level of detail specification is normally documented in the form of a set of examples.

Scope and level of detail decisions are tentative at this stage of the project and should be updated as the description data becomes available. An astute project leader will regularly assess the adequacy of the description data captured against the specified needs and information goals of the client.

Organize for Data Collection

Once the initial project purpose and context have been determined, the task of organizing for data collection can begin in earnest. At this point, the makeup of the

project team will be solidified, team member roles will be established, and scenario development responsibilities will be assigned to team members.

The following roles are normally assumed by personnel involved in an IDEF3 process flow description capture process.

1. **Analyst:** The IDEF3 expert who will be the primary developer of the IDEF3 process flow description.

2. **Client:** The person or organization requesting the description development.

3. **Domain Expert:** The knowledge source person in the application domain of interest.

4. **Primary Contact:** The individual who acts as the interface between the analyst and the domain expert.

5. **Project Leader:** The person ultimately responsible for the entire description development effort.

6. **Reviewers:** Persons knowledgeable of the domain and/or the IDEF3 method who are responsible for reviewing and approving draft descriptions and documents. Reviewers authorized to make written critiques of IDEF3 schematics are *commentors*. The remainder are *readers*. Both team members and domain experts can be reviewers (see Section 4).

7. **Librarian:** A person assigned the responsibility of maintaining source material logs and files of documents, making copies, distributing IDEF3 kits, and keeping records.

8. **Team Members:** All personnel involved with the IDEF3 process flow description development project.

Among the roles assigned to team members is that of the project librarian. With large systems, the role of the librarian is essential. In smaller efforts, that role may be assumed by the analyst. In establishing the librarian function, the project leader assigns an individual(s) to be responsible for collecting, cataloging, controlling, and distributing source material, IDEF3 kits, glossaries, files, and so forth throughout the project. Additionally, the librarian function is responsible for assembling reference models and materials from external sources (e.g., process benchmarks in industry) that can be used to accelerate team efforts. A glossary of terms may also be maintained by the librarian as a reference to ensure that analysts understand terminology that is unique to a discipline, industry sector, company, or company segment. Whether maintained by the librarian, or

informally shared among analysts, the glossary of terms will grow and undergo incremental refinement throughout the project.

A pivotal task in organizing the data collection effort is identifying the key sources of knowledge and information in the domain. Working with the primary contact, the project leader or analyst compiles a list of experts to be interviewed. In compiling this list, it is helpful to obtain background information about each expert from the primary contact. This includes information about the responsibilities, current assignments, and other areas within or related to the domain in which the expert has experience. The name, location, and telephone number for each expert should also be recorded.

Throughout the data collection effort, other valuable sources of information will be sought and identified. Some of these might include operating instructions, procedure manuals, employee handbooks, regulations, policy manuals, project files, reusable IDEF models, and models derived through the use of other methods and techniques.

In addition to organizing the structure of the team, the project leader also needs to organize the activities of the team. Organizing process description capture activity may begin by casting the general IDEF3 procedure into a more formalized method application guide tailored to the specific needs of the project. A method application guide outlines a project-specific application of the IDEF3 procedure tailored to meet the needs of the effort. Among the items that may be included in the method application guide are modeling conventions to be used, standard outlines for interviewing domain experts, method and tool interface specifications, project library use procedures, and a standard glossary of terms. This guide may be accompanied by a project plan. A typical project plan will delineate phases of effort with clearly established tasks and milestones, intermediate and final deliverables, individual team member assignments, informal and formal reporting structures, and so forth.

Collect and Analyze Data

At this point, the stage is set for actual data capture. The main information sources available to the team are domain experts and source documents in the organization. The analyst must work closely with domain experts to effectively capture data relevant to the description development effort.

The data collection process is both iterative and interactive. Preliminary data provides guidelines for organizing the knowledge acquisition effort. Analysts interact with domain experts to obtain initial descriptions, both written and verbalized, of the process under study. The names of the activities and participating objects are extracted from these initial descriptions. Often, it is necessary to interview different experts who are knowledgeable about different aspects of the process. It is also often necessary to

conduct follow-up interviews and multiple kit reviews with domain experts. The data gathered through this process must be carefully recorded so that the final description can be easily consolidated as an accurate reflection of domain expert observations.

Prepare for Interviews

No specific format for data collection is prescribed by the IDEF3 method. However, before the interview, the analyst should prepare a tentative agenda and some specific questions. Analysts are encouraged to prepare a brief outline of: (1) the purpose of the interview with the expert, (2) the topics to be covered, (3) the types of information being sought, (4) the authority for requesting the interview, and (5) questions that can be used to motivate discussion. On large projects, project leaders may wish to include more formalized interview preparation guidelines and standards in a method application guide—including standard interview planning sheets, question templates, glossaries of terms, and so forth.

A number of activities contribute to successful interview preparation, each of which is left to the discretion of the analyst as dictated by the needs of the project and the constraints involved. In general, the following activities are accomplished prior to the interview:

1. Schedule the interview and make necessary logistics preparations.

2. Establish the goal(s) of the interview.

3. Prepare candidate questions.

4. Anticipate the probable questions and concerns of the person being interviewed and be prepared to resolve those concerns.

Additional interview preparation activities may also be needed or desirable. For example, analysts may wish to analyze previously collected documents describing the client's formal system or process, and prepare IDEF3 descriptions from those documents as a launching point for discussion. Similarly, analysts may use benchmark models of similar systems to afford the opportunity of interactively working with the domain expert to identify similarities and differences.

Once a list of experts to be interviewed has been compiled, an interview schedule can be developed. Interviews are normally scheduled with domain experts through the primary contact. Whether done through the primary contact or by more direct means, the analyst should make sure that the scheduled time and duration of the interview is coordinated with the person being interviewed and his or her supervisor.

Additional logistics considerations are also important to the success of the interview, such as finding and reserving a suitable location to conduct the interview and arranging for the necessary supplies. Analysts also generally find it useful to plan the attire they wear to the interview in order to convey a professional appearance and still set the interviewee at ease.

The goal(s) of the interview should be established up front. In establishing the interview goal(s), analysts state why the interview is being scheduled and what information is minimally required from the domain expert. Preparing a goal statement is often helpful if it is kept as succinct as possible so as to provide a general direction for the interview line of questioning.

Once the goal(s) of the interview has been established, candidate questions can be formulated. Candidate questions should be written down and organized into a logical sequence. With experience and practice, analysts will eventually become proficient in developing questions that are clear, that use words and phrases appropriate to the educational level and cultural background of the person being interviewed, and which invite rather than lead answers. Although the analyst should be cautious not to over prepare, the exercise of writing questions down and analyzing the way they are formed helps develop good interviewing skills. The time invested to this activity must be balanced, however, against the possibility that the questions formulated may or may not actually be used. Their necessity may be eliminated through the discovery of new information; or, the interview may go down a line of discussion that was not previously anticipated.

Several preparatory items are often overlooked. Analysts need to provide the person being interviewed with the information necessary to understand why they are being interviewed, what will be done with the information they provide, and what they can expect in return. Each interview, and particularly the first, should begin by establishing a mutual understanding of these items before attempting to satisfy the information needs of the analyst. The following list is representative of the questions and concerns the analyst should be prepared to address (Harrington, 1991).

1. Why is the interview being conducted?
2. Who authorized the interview?
3. Who else is being interviewed?
4. How was the interviewee selected and by whom?
5. How will the information be used?
6. Will the interviewee remain anonymous?

7. Will the interviewee be quoted in summary findings?

8. What feedback will the interviewee receive?

9. How might the interviewee participate in the outcome of the process?

10. What does the interviewee stand to gain?

11. Why is highly detailed, accurate information important to the success of the interview and the project?

12. How does the interviewee play a key role in an important process?

Interview Domain Experts

Interviews may be conducted throughout the project to collect additional information, to clarify previous information, or to validate IDEF3 models with the domain expert.

The interview with the expert is critical. The analyst (interviewer) should create a positive, and friendly atmosphere during the interview. The interviewer should attempt to convey to the domain expert the feeling that they are working together to create the required description and to solve some problem for the organization. Analysts should constantly remind themselves that the expert is the one with the knowledge of how a process should or does work. Generally, the expert is interested in helping and will often provide questions and lines of investigation that the interviewer had not thought of pursuing.

The expert often provides copies of documents and forms used in the *current* process. This documentation may actually outline the process flow, or rather, the «Should-Be» process flow. The interviewer must remember that the main focus must be on the process actually performed, rather than formally documented procedures that may or may not be followed. When focused on how the process is performed today, analysts should be cautious to avoid talking about the «To-Be» system to avoid introducing bias in the domain expert's answers.

Collect Names of Objects

Under normal circumstances, one of the first types of information an expert provides are the names of objects involved in the domain. The interviewer should carefully note these objects. During the analysis following the interview, the analyst/interviewer will prepare a list of all these objects. This list (object pool) will be analyzed later to associate the objects of the domain with the UOBs that are relevant to the domain.

Collect Activity Names

The named activities provided by the expert should be carefully noted. These will often become the names of UOBs that will be arranged to form process schematics. As the names of the activities are collected, some notion of their sequencing and structure should be determined and noted. During the analysis that follows the interview, the analyst/interviewer will prepare a list of all these activities. This list is referred to as the *pool* of potential UOBs for the IDEF3 schematics.

Collect Facts and Constraints Related to Process Occurrences

Facts relative to UOBs and objects and constraining relations between objects, facts between objects and UOBs, and facts between UOBs should also be noted during the interview. The types of information to focus on during the interview include:

1. Constraints that govern the initiation of a process.
2. Conditions that must hold during the process.
3. Conditions that signal the termination of a process.
4. Processes triggered by the initiation or termination of the process.
5. Properties of an occurrence of the process (e.g., duration, interruptability).
6. Objects that participate as agents, information, resources, or products in the process.
7. Properties of the objects (e.g., particularly those associated with the process such as arrival rates or spoilage rates).
8. Relations or associations between the objects in a single process.
9. Relations or constraints on objects between processes (e.g., shared resources).
10. Conditions that must be satisfied relative to the objects participating in the process.
11. The distinction between normal and exceptional situations in the occurrence of a process.

Collectively, this set of information is referred to as *facts*.

Collect Situation Descriptions

In IDEF3, we refer to a situation description as the characterization of an occurrence of a process. This characterization includes the association of activities with the collection of objects standing in particular relations during an occurrence. It also includes the association of an activity with the other activities that precede or follow its occurrence. Situation descriptions often can be obtained by observing the process in action (e.g., visiting the factory where a particular part is made). However, such direct observation generally only provides information on the normal processing of short-duration situations. Generally, the analyst must rely on the domain expert to provide special insight, into both the normal processing of long-duration situations and the processing of exceptions to the norm. During the analysis of these situation descriptions, the analyst will add to the lists of objects and activities previously discovered. Analysis of the situation descriptions will provide the necessary insight into the sequencing of activities, the list of facts, and the constraints associated with the process to be described.

Collect and Catalog Source Material

As appropriate, analysts should request to see information artifacts of the process (e.g., forms, screens) that are included in the domain expert's description. To the extent practical, copies of these information artifacts should be collected for further analysis. All data collected during the course of the project should be logged on an IDEF3 Source Material Log as illustrated in Figure 4-2.

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Example IDEF3 Description NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

Source Material No.	Source Material Name	Collected From	Collected By	Date Collected
SM1	Purchase Requisition/Form PI-R6 4-72	U.R. Buyer		
SM2	Procedure #079-003 /Rev. 00 "Preparation of the Requisition"	U.R. Buyer		
SM3	Procedure #079-001/ Rev. 00 "Preparation of the Purchase Order"	Policy and Procedures Manual		
SM4	Procedure #101-506 "Purchasing Codes"	Policy and Procedures Manual		
SM5	B.J. Commodity Code List	U.R. Buyer		
SM6	B.J. Product Code List	U.R. Buyer		
SM				
SM				
SM				
SM				
SM				

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE: Source Material Log

Figure 4-2
Source Material Log

For each source material item referenced in the Source Material Log, there is a Source Material Description that is used to record more detailed information. Figure 4-3 provides an example of this form.

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Example IDEF3 Description NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

Source Material No. SM	Source Material Name: Supports: Comments: Abstract:
Source Material No. SM	Source Material Name: Supports: Comments: Abstract:

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE Source Material Description

Figure 4-3
Source Material Description Form

Each entry on the Source Material Description Form is identified by the source material number and name to which the entry corresponds. This enables traceability to the source material from which candidate process description elements are individuated by member(s) of the description development team. Additional fields included on the form include the following:

1. **Supports**: The IDEF3 element numbers (e.g., UOB numbers, Object State numbers) supported by the source material are documented in this field, providing traceability for description elements to specific source material.

2. **Comments:** This field is used to record special features or comments worth referencing at a later date about the item being cataloged.

3. **Abstract:** The abstract provides a concise overview of the main concepts discussed in the source material.

Analyze Collected Data

Following data collection, interview notes are analyzed, source material is studied and logged in the Source Material Log, and the initial findings are cataloged into lists called pools. Four pools are used in IDEF3: (1) object pool, (2) scenario pool, (3) UOB pool, and (4) object state pool. Figure 4-4 shows an example of an object pool. All other IDEF3 pools use the same basic layout as illustrated in Figure 4-4.[10]

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Example IDEF3 Description NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

Object		Source Material No.	Object		Source Material No.
ID No.	Name		ID No.	Name	
01	Purchase Requisition	SM1	018	Bill of Material	SM38
02	Buyer	SM2	019	Route Sheet	SM40
03	Vendor	SM3	020	Destination	SM41
04	Purchase Order	SM15	021	Approver	SM43
05	Ship To Location	SM6			
06	Requester	SM11			
07	Department	SM12			
08	Pattern	SM21			
09	Part	SM26			
010	Purchase Req. Item	SM23			
011	Commodity	SM30			
012	Purchase Req. Item	SM31			
013	Job	SM34			
014	Account	SM36			
015	Product	SM37			
016	B.M. Page	SM38			

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE: Object Pool

Figure 4-4
Example IDEF3 Object Pool

Acquire Additional Information as Required

Both analysis of the data collected and initial attempts to construct IDEF3 schematics often reveal the need for additional data. A number of approaches are available for collecting additional information, including:

[10] The field labeled «Object» on the form would be changed to Scenario, UOB, or Object State, as appropriate. In addition, the ID No. fields would be prefaced by S (for Scenario), UOB, or OS (for Object State) to compose the element identification numbers.

1. Conducting follow-up interviews to answer questions and/or identify additional source material.

2. Using the kit review process.

3. Arranging for direct observation of the process or scenario in question.

4. Revisiting source material with a new focus of analysis.

5. Conducting facilitated workshops.

The approach or combination of approaches used will be determined by both the nature of the information needed and the purpose for which IDEF3 is being used. Generally speaking, workshops are used only for group brainstorming and consensus building, such as when developing ideas for and consensus among alternative «To-Be» process designs.

Formulate IDEF3 Schematics

Two types of IDEF3 schematics provide a mechanism for collecting and displaying process description information. The Process Schematic provides a process-centered view of a scenario. The Object Schematic provides an object-centered view of a scenario or set of scenarios. Both constitute graphical projections of domain expert descriptions.

A key point to remember in constructing IDEF3 schematics is that schematic development should not be constrained by idealized, testable conditions that must be satisfied, short of simple accuracy. For example, it is quite normal for Process Schematics to initially not show a logical flow. These schematics often start out with a set of UOB boxes with little connectivity among them. This may be because the complete picture has not yet been acquired. Descriptions, after all, constitute a recording of facts or beliefs about something within the realm of a domain expert's knowledge or experience. Such descriptions are generally incomplete; that is, the person giving a description may omit facts that he or she does not think are relevant, which he or she has forgotten in the course of describing the system, or of which he or she has no knowledge.[11] Incredible as it may seem, there are many systems that work which have elements that no single person understands or even knows about.

---

[11] When domain experts know that some activity, object, or relation exists and they know what that is, they can easily display that knowledge using the IDEF3 schematic elements. When domain experts know that one of these things does not exist, that fact is also easily captured. Specialized syntactic conventions may also be established to explicitly document a domain expert's knowledge that something exists of which they have no knowledge (i.e., Socratic versus Platonic information). For example, a squiggly-lined arrow extending between UOBs might be used to indicate that some

The fact that a schematic includes elements that are disconnected should not cause overwhelming concern. It is not uncommon for the project to end successfully while there are still gaps in several of the schematics. This can happen when the goals of the project do not require expenditure of the necessary effort to fill those gaps. Furthermore, when using IDEF3 to capture descriptions of the current environment, the IDEF3 users is not *designing* a system but rather organizing *known* facts about how a system works. The resulting descriptions may serve as the raw material from which models are made. Thus, in addition to providing a precise and well-structured mechanism to capture and store process knowledge, IDEF3 descriptions can be reused to construct multiple idealizations or models with which to simulate and predict system behavior.

Formulate Process Schematics

Process Schematics provide a *process-centered* view of a process. These schematics organize process knowledge with a focus on processes and their temporal, causal, and logical relations within the context of a scenario.

The steps involved in constructing a Process Schematic are as follows.

1. Identify the UOBs.
2. Associate the UOBs with the appropriate scenario.
3. Identify precedence constraints between pairs of UOBs in a scenario and layout initial schematic.
4. Add junctions for logic description.
5. Add constrained precedence links as required.
6. Develop elaborations for UOBs, junctions, and links as needed.
7. Develop decompositions for selected UOBs.
8. Add relational links to highlight additional relationships of interest.

Identify the UOBs

Having completed initial data gathering activities, the analyst should have lists of objects of interest, activities, facts, and constraints. Using this data and the situation

---

unknown collection of UOBs exist between those indicated. A state represented in the shape of a cloud might be used to indicate knowledge of the existence of a state, although what that state is may not be known. When used, however, such conventions should only be used during intermediate stages of description development.

descriptions, the analyst identifies the UOBs and begins to formulate the general structure of the IDEF3 schematics.

The initial foundations for this activity occurred while establishing the context of the project, wherein the analyst will have identified the scenario(s) of interest. At least one Process Schematic will typically be developed for each scenario identified. The scenario serving as the context for a given Process Schematic establishes the scope of the analyst's search for candidate UOBs.

Identifying UOBs is a mental classification activity that uses knowledge of the IDEF3 paradigm and facts collected from the domain expert to identify and refine a candidate set of UOBs. It is helpful in this process for analysts to be attuned to how people describe processes. The capture of a description of «what's going on» within an organization or any complex system needs to account for a number of natural language concepts. Each of the following concepts is used in everyday language to describe «things that happen in the world.»

1.	Function	4.	Activity	7.	Action
2.	Process	5.	Operation	8.	Event
3.	Scenario	6.	Decision	9.	Procedure

Each of these concepts involves some circumscribed behavior. For instance, a reference to the Planning Activity, Make-or-Buy Decision, or the Contract Award Event carves up the world into spatio-temporal chunks to allow a description of «what is going on» in that chunk to be separated from the rest of the world. In IDEF3, a generic packet of information (or UOB) encapsulates concepts such as those listed above.

Lists of candidate scenarios developed in the project definition stage may be an early source for candidate UOBs. For example, by analyzing the initial set of candidate scenarios used to scope the project, analysts may question whether the scenarios may be «threaded together,» or subsumed by another candidate scenario listed.[12]

The initial fact set is also a valuable source of candidate UOBs. Identifying candidate UOBs may begin by searching for named processes (e.g., acquisition process), imperative verb forms (e.g., budget funds), and gerunds (e.g., identifying operational needs) in the domain expert's description. Through this analysis process, both additional scenarios and candidate UOBs may be identified. Unlike scenarios, which tend to be more general, UOBs are more specific, with definite time constraints. That is, if one is able to identify strong causal or time-ordering dependencies, it is probable that a UOB rather than a

[12] Some candidate scenarios listed may also be recognized simply as alternative names for the same situation type.

scenario has been identified. Thus, phrases like, «Such events may require redefinition of assigned tasks *in response to* shifts in national security policy,» may yield two candidate UOBs (*Redefine assigned tasks* and *Revise national security policy*) that stand in the causal relation *in response to*.

People also make extensive use of objectification and metonomy to describe processes without names. That is, most processes, like most objects, do not have names. Objectification of a process simply means that one takes a process and describes it as an object. For example, the process *Acquire Weapon System* may be referred to by a member of Congress as *weapon system acquisition*, or simply *system acquisition*. Metonomy is a form of expression that uses the name of an important object or relationship (e.g., attribute) that is associated with the thing being described as a definite descriptor for that thing. For example, processes may be referred to by the name of the principal product produced (e.g., mission needs determination, Milestone 1 decision).

Recognizing these forms of expression, the analyst can quickly identify candidate UOBs, associate them with the appropriate scenarios, and begin the process of constructing the initial Process Schematic.

Layout Initial Process Schematic

An initial Process Schematic is developed to illustrate the analyst's understanding of the information collected from the expert. Using the initial schematic, the analyst reviews the description with the domain expert to ensure the description is correct. These initial schematics also assist the domain expert in recalling additional experience. The process of initial data collection is limited by the ability of the domain expert to recall his or her internalized knowledge.

Obtaining a reasonably accurate and complete description from an expert is an iterative process that must be repeated until the analyst's schematic agrees with the domain expert's knowledge. In some situations, it may be possible for the analyst and the domain expert to develop the descriptions together, rather than developing a draft description followed by a review procedure. The joint development approach can reduce the development time and produce descriptions that are more complete the first time.

A Process Schematic Summary form (See Figure 4-5) aids the analyst in coordinating review activity with domain experts and developing the Process Schematic from the raw data. A textual description, or glossary of the Process Schematic, is part of this form. This text should contain a statement of the purpose for the schematic and may contain other information that does not readily fit into the other fields. In addition to the textual description, the analyst records the UOBs and the other IDEF3 elements (UOBs, Scenarios, and Transition Schematics) that are referenced in the schematic. Initial

completion of this form is part of the analysis activity associated with constructing the Process Schematic.

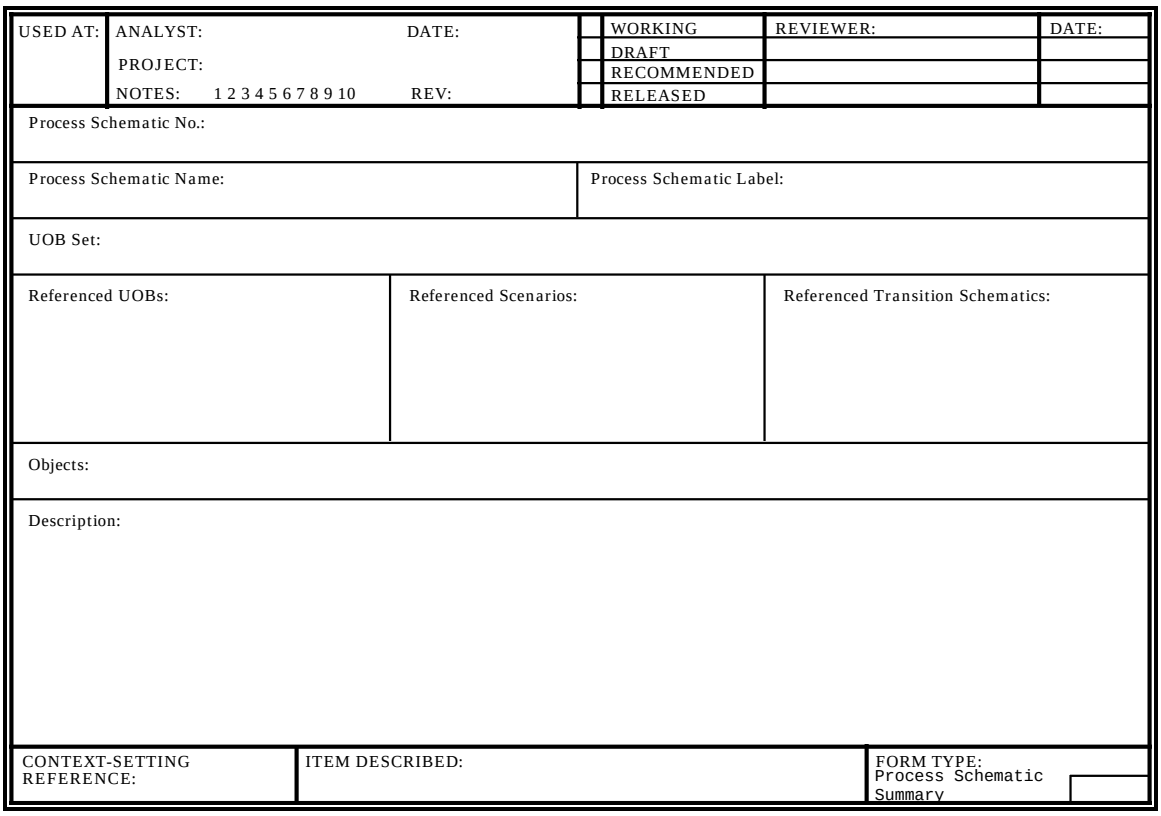

Figure 4-5
Process Schematic Summary Form

Develop Elaborations for UOBs, Junctions, and Links

An IDEF3 Process Schematic graphically describes a process in terms of the UOBs that occur in it, with each relevant UOB in the process represented by a UOB box. However, a cursory inspection of the UOB boxes in a schematic will not provide a complete picture of the processes being described. Elaborations provide detailed characterizations of IDEF3 elements in the schematic. Information on the elaboration forms provide the most detailed characterization of the expert's description. The schematic is the graphical presentation of a portion of this information. For most purposes, natural language statements suffice for IDEF3 element elaborations. However, when the purpose for the schematics requires more structure and precision than natural English statements in the elaboration, users can use IDEF3's elaboration language (See Appendix A).

Elaborations for UOBs, junctions, and precedence links are developed from the interview data and reviewed by the domain expert whose knowledge the description represents. Initially, these elaborations may look like simple glossary entries. However, as the data analysis progresses, the elaborations become more structured and concise. Typical information found in an elaboration include participating objects and their roles, facts of interest, and constraints. These natural language elaborations will be written up on elaboration forms (See Figures 4-6 through 4-9).

A UOB elaboration document includes: (1) the UOB's name, label, and UOB number; (2) listings of the *object types and instances*, *facts*, and *constraints* that determine the nature and structure of the UOB; and (3) a textual description of the UOB (See Figure 4-6).

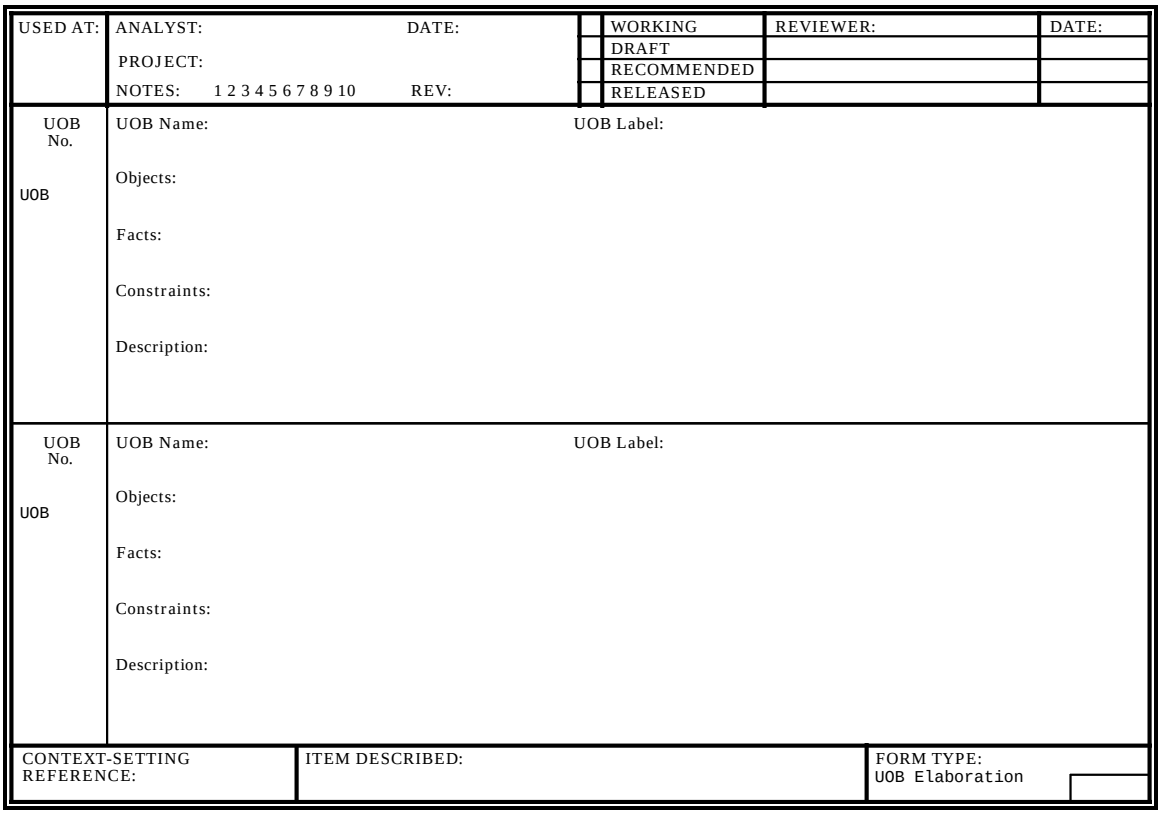

Figure 4-6
UOB Elaboration Form

The following list contains a description of the contents of each field on the UOB elaboration document.

1. **UOB No.:** This section contains the UOB Number of the associated UOB. The UOB number uniquely identifies the UOB box associated with the elaboration document.

2. **UOB Name:** This section contains the UOB Name.

3. **UOB Label:** This section contains the UOB Label (i.e., the UOB Name, some part of the Name, or an abbreviation displayed in a UOB box).

4. **Objects**: This section lists the names of all the objects (types or instances) which participate in the process being described by the UOB. These objects can be either physical or conceptual. Objects can be created, modified, or destroyed during the process. It may be useful to categorize an object as an agent, affected, a participant, or a created or destroyed object.

    a. *Agent* – if the objects (or objects of the type in question) play an active causal role in instances of the UOB.

    b. *Affected* – if the object (or instances of the type) is changed during instances of the UOB activity.

    c. *Participant* – if no causality or transformation is associated with the object (or instances of the type) in instances of the UOB.

    d. *Created* or *Destroyed* – if the object (or instances of the type) are created or destroyed in instances of the UOB.

5. **Facts:** This field lists facts about instances of the UOB.

6. **Constraints:** This field lists constraints on the UOB, i.e., facts about what must hold in all instances of the UOB.

7. **Description:** This field contains a glossary entry (textual description) for the UOB. Typically, the glossary entry provides a textual recount of the information already in the object, fact, and constraint lists.

To facilitate capturing the decision logic of a junction, an elaboration document can be attached to a given junction in a Process Schematic, as illustrated in Figure 4-7.

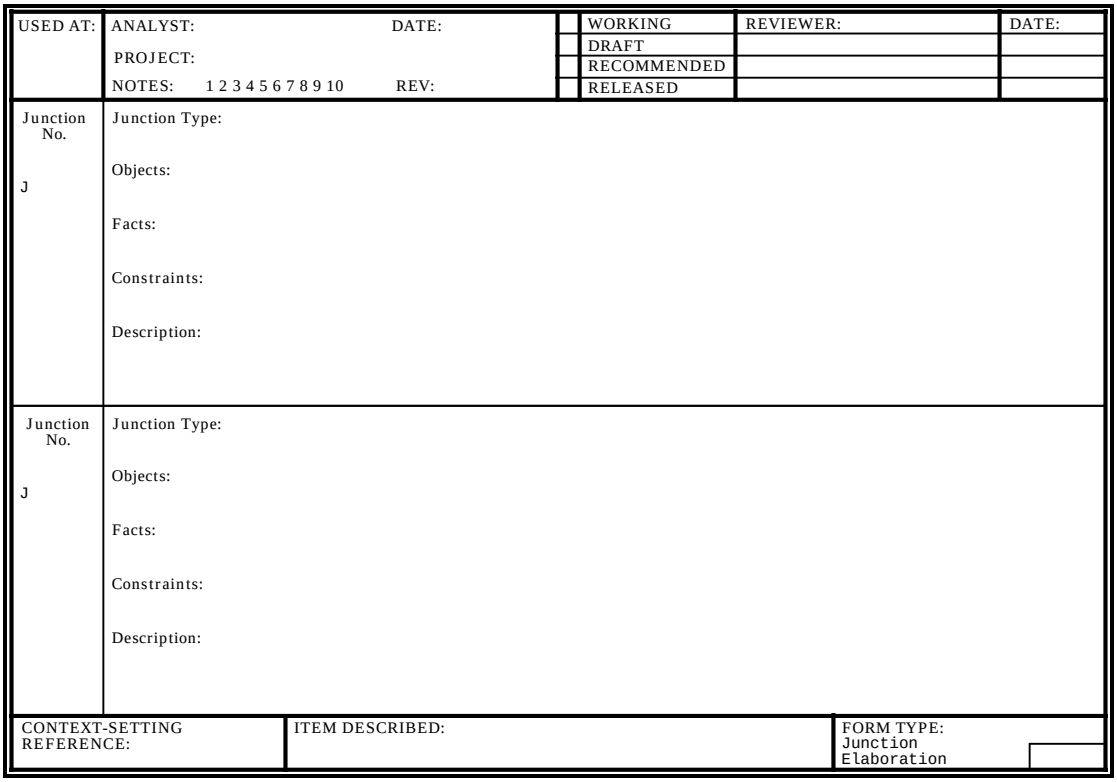

Figure 4-7
Junction Elaboration Form

The core fields in a junction elaboration document are as follows.

1. **Junction No.:** The junction number, prefaced by the letter «J» (for junction), that uniquely identifies the junction within the description.

2. **Junction Type:** Asynchronous AND, asynchronous OR, synchronous AND, synchronous OR, or XOR (exclusive or).

3. **Objects:** All significant objects (types or instances) associated with the junction. Typically, these objects are the agents that enforce junction constraints.

4. **Facts:** Noteworthy, nonconstraining facts associated with the junction and, in particular, facts involving the objects associated with the junction.

5. **Constraints:** A specification of the decision logic and any other constraints associated with the junction. Ideally, this specification will be given in a logically precise form in the IDEF3 elaboration language. The elaboration language is defined, discussed, and illustrated in Appendix A.

6. **Description:** An informal description of the decision logic, along with any other useful annotations or background information.

The special constraints indicated by constrained precedence links are recorded in a precedence link elaboration document (see Figure 4-8). This document is similar to a UOB elaboration in format and purpose.

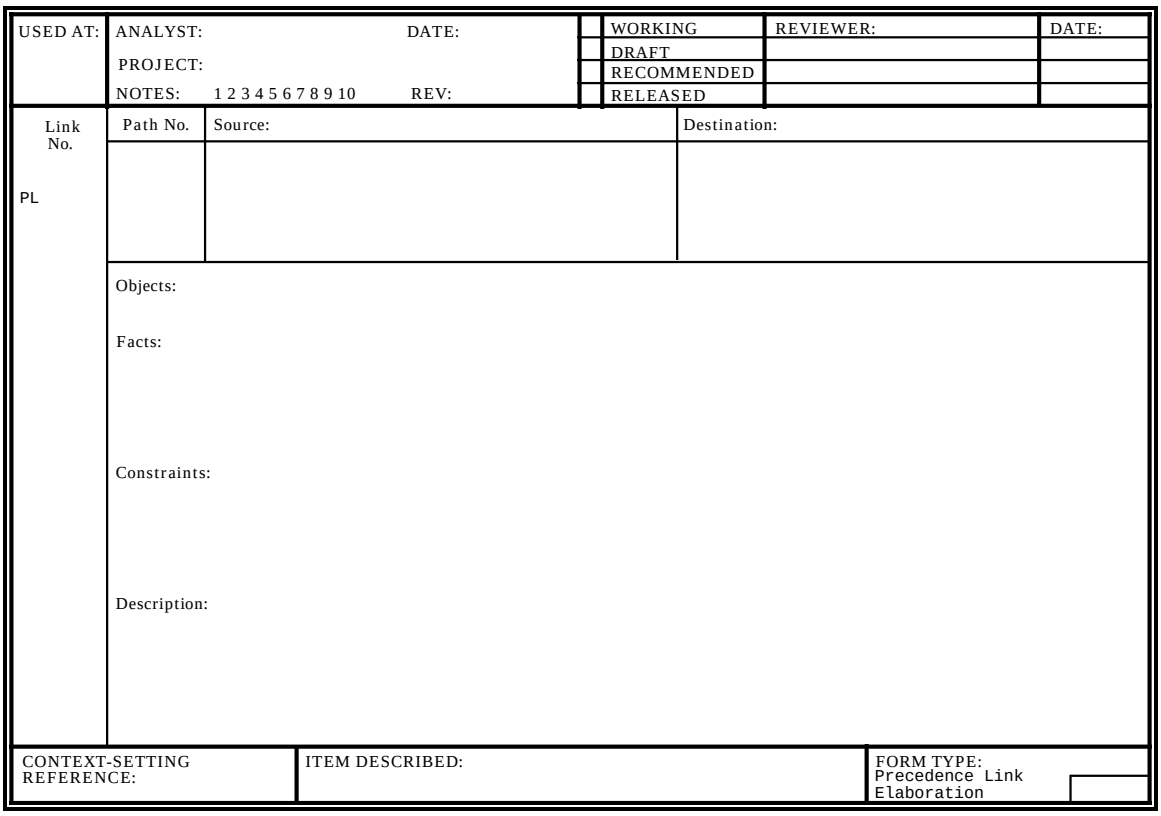

Figure 4-8
Precedence Link Elaboration Form

The following describes the main content of a precedence link elaboration document.

1. **Link No.:** A link number, prefaced by the letters «PL», that uniquely identifies the precedence link within the description.

2. **Path No.:** A link path number, comprised of the Link No. and a unique integer, separated by a period. For example, given a precedence link PL1 that separates into three alternative paths following a junction, the path link numbers would be PL1.1, PL1.2, and PL1.3.

3. **Source:** Name of the source IDEF3 element (i.e., UOB or referent) of the link in the specified path.

4. **Destination:** Name of the destination IDEF3 element (i.e., UOB or referent) of the link in the specified path.

4. **Objects:** All significant objects (types or instances) that participate in the relation that the link represents. Typically, these objects are constituents in the source or destination of the relation indicated by the link.

5. **Facts:** Noteworthy, nonconstraining facts involving the objects that participate in the relationship represented by the link.

6. **Constraints:** Noteworthy constraints that hold between the source and destination UOBs or between some of their constituent objects. This field contains, in particular, the constraints indicated by the general constrained precedence link.

7. **Description:** The descriptive glossary associated with the link. Any descriptive information that does not fit logically into any of the other fields in the document is placed here.

As appropriate, objects, facts, and constraints that are uniquely associated with a particular link path will be so identified.

Special constraints indicated by dashed links are recorded in a dashed link specification document (see Figure 4-9).

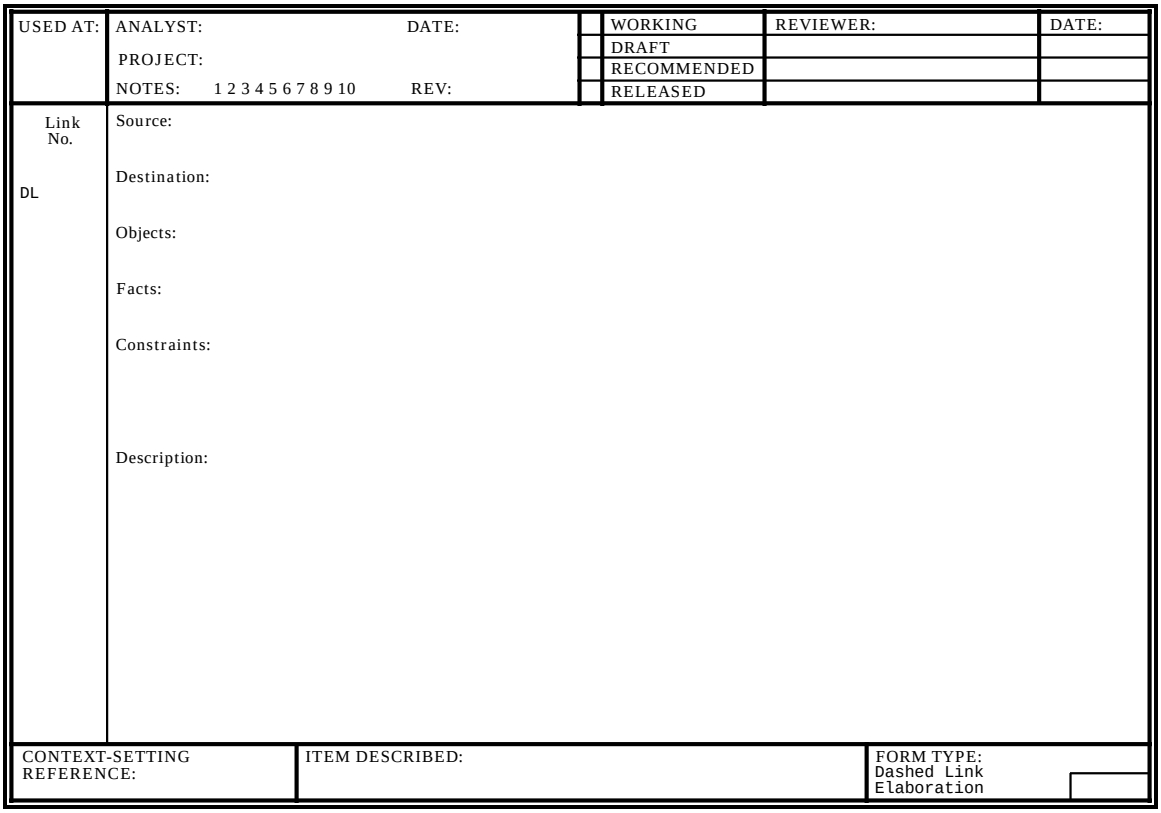

Figure 4-9
Dashed Link Elaboration Form

The following describes the main content of a dashed link elaboration document.

1. **Link No.:** A link number, prefaced by the letters «DL» (for dashed link), that uniquely identifies the dashed link within the description.

2. **Source:** Name of the source IDEF3 element (e.g., UOB, referent) of the relation indicated by a link.

3. **Destination:** Name of the destination IDEF3 element (e.g., UOB, referent) of the relation indicated by the link.

4. **Objects:** All significant objects (types or instances) that participate in the relation that the dashed link represents. Typically, these objects are constituents in the source or destination of the relation indicated by the dashed link.

5. **Facts:** Noteworthy, nonconstraining facts involving the objects that participate in the relationship represented by the dashed link.

6. **Constraints:** Noteworthy constraints that hold between the source and destination elements or between some of their constituent objects.

7. **Description:** The glossary associated with the dashed link. Any descriptive information that does not logically fit into the other fields in the document may be placed here.

Develop Decompositions for Selected Units of Behavior

A decomposition of a UOB is a collection of other UOBs that provides additional details of a process represented by the parent UOB from a particular perspective. UOBs at the scenario level will usually have decompositions. The UOBs in these decompositions may also be decomposed. Different decompositions normally result from different domain expert views of what happens during an activity. They can also result from abstracting some participating object's view of the process. For example, a decomposition view might be created to show the processing steps required of the information system in order to support an organizational activity. Finally, decompositions can be produced by the analyst for selected UOBs to simplify a schematic. Decompositions are schematics providing a more detailed view or different perspectives of a process with a clearly defined viewpoint. Decompositions are often developed to capture alternative views of a process or to simplify a process description schematic.

Like IDEF3 description development, the decomposition development process is a refinement process. Decomposition development follows the same procedure as that of the primary description development. This refinement cycle consists of activities to (1) analyze the activity, (2) collect additional data, (3) describe situations in terms of related UOBs, (4) review, and (5) if necessary, return to a previous step in the procedure.

Formulate Object Schematics

Object Schematics are provided in IDEF3 to complement Process Schematics. Object Schematics enable an *object-centered view* of the process being described by facilitating detailed characterization of objects, object states, state transitions, and inter-object relations. Object Schematic development may occur before, during, or after the development of Process Schematics. This section provides guidelines for developing Object Schematics.

The steps involved in constructing an Object Schematic are as follows.

1. Select objects of interest.

2. Identify object states.

3. Characterize possible transitions between states and lay out the basic state transition schematic.

4. Add junctions, as required, to reflect alternative state transition paths and object composition logic.

5. Attach referents for participating UOBs, scenarios, and Object Schematics to appropriate points on the schematic.

6. Develop elaborations as needed.

7. Develop object state decompositions for selected object states.

8. Identify and mark transitions yielding the same object.

9. Add other objects and relations to the schematic as needed to provide useful context-setting information.

Select Objects of Interest

The first task in constructing the Object Schematic portion of a description is deciding which object(s) to describe. Basically, the analyst must identify which objects play an important role in the domain expert's knowledge about the system. The list of objects involved in a process may be extensive. In comparison, the list of objects of special interest is likely to be small. These are generally objects that are modified by the process being described. Since Object Schematic creation normally follows the development of one or more Process Schematics, a primary source for the objects of interest will be (1) UOB elaborations, (2) scenario descriptions, (3) models of the information required by the scenario (e.g., other IDEF models), and (4) original interview data. Regardless of the source of the objects, they have two features in common: (1) they undergo noticeable changes in the process and (2) they exist in several states at various points in the process.

Because an object theoretically can be *any* physical or conceptual thing, there is no scientific method to decide which objects are in a domain. However, as a general heuristic, in IDEF3 we are interested in objects that play an important role in the operation of the system. Such objects will normally be named; that is, the analyst will find a word or phrase that appears frequently in the interview information. Whatever this word or phrase refers to can be considered a possible object for consideration. The second issue to consider is whether the objects of interest have states of interest. Again, some of the heuristics are: (1) each object state should display characteristics commonly recognized in the domain; (2) the object should be recognized to exist in a state for a period of time; and (3) there are recognized constraints or process that enable, cause, or inhibit the state changes. For each selected object, at least one Object Schematic is developed.

Layout Initial Object Schematic

For each Object Schematic, the creation of an Object Schematic Summary form is necessary (See Figure 4-10). A textual description, or glossary of the Object Schematic is part of this form. This text should contain a statement of the purpose for the schematic and will generally contain other information about the Object Schematic that does not readily fit into the other fields (e.g., ontology information that would later be included in an IDEF5 model). In addition to the textual description, the analyst records the object states and the other IDEF3 elements (UOBs, Scenarios, and Transition Schematics) that are referenced in the schematic. Initial completion of this form is part of the analysis activity associated with constructing the Object Schematic. This initial work aids the analyst in developing an Object Schematic from the raw data.

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Example IDEF3 Description NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

Object Schematic No.:	Object Schematic Type:
Object Schematic Name:	Object Schematic Label:
Object State Set:	

Referenced UOBs:	Referenced Scenarios:	Referenced Transition Schematics:
Objects:		
Description:		

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE: Object Schematic Summary

Figure 4-10
Object Schematic Summary Form

The following list contains a description of the fields contained in a Transition Schematic description form.

1. **Object Schematic No.:** A unique identification number for the Object Schematic, prefaced by the letters «TS» for transition schematics and «OBS» for Object Schematics.

2. **Object Schematic Type:** The Object Schematic type may be described as a Transition Schematic, Enhanced Transition Schematic, or an Object Schematic. Transition Schematics (i.e., Transition and Enhanced Transition Schematics) are special types of Object Schematics whose context is established by a single scenario.

3. **Object Schematic Name:** The name of the Object Schematic.

4. **Object Schematic Label:** The Object Schematic Name, some part of the Name, or an abbreviation used for convenience when displayed in an IDEF3 graphical element (e.g., when displayed in a referent).

5. **Object State Set:** The set of object states that make up the state transition represented by the Object Schematic, if the schematic is a type of Transition Schematic. If the schematic is a general Object Schematic, this field lists the object states included in the Object Schematic.

6. **Referenced UOBs:** The set of UOBs referenced by the Object Schematic.

7. **Referenced Scenarios:** The set of scenarios referenced by the Object Schematic.

8. **Referenced Transition Schematics:** The set of Transition Schematics referenced by the Object Schematic.

9. **Objects:** The set of objects included in the Object Schematic for context-setting purposes.

10. **Description:** A textual description, or glossary, associated with the Object Schematic. Any descriptive information that does not logically fit into the other fields in the document may be placed here.

The next step in Object Schematic development is to describe each object state and characterize the state transitions. To accomplish this, the analyst will perform the following tasks:

1. Identify the defining characteristics for each object state.

2. Identify the conditions for leaving each state.

3. Identify the criteria for entering each state.

4. Identify special conditions for enabling an object in the state to attempt a transition.

5. Identify the possible transitions between states.

6. Identify the activities that cause, allow, or are caused by each transition.

Develop Elaborations for Object States, Objects, Junctions, and Links

The results of the first two activities are recorded on the object state elaboration form for each affected state. The results of the third and fourth activities are documented on the transition link elaboration form. The results of the last three activities determine the schematic layout. The structure and content of these elaboration forms closely parallel those associated with Process Schematic elements. Examples of these forms are provided in the following few pages (See Figures 4-11 through 4-14).

The object state elaboration document is used to capture the elaborations of the object states that participate in the state transitions depicted in an Object Schematic. An object state elaboration document is constructed for every object state represented in the Object Schematic. In addition to enabling a detailed characterization of a state, the object state elaboration document form carries information about state and exit conditions, as discussed in Section 3, IDEF3 Process Description Language. The object state elaboration document is shown in Figure 4-11.

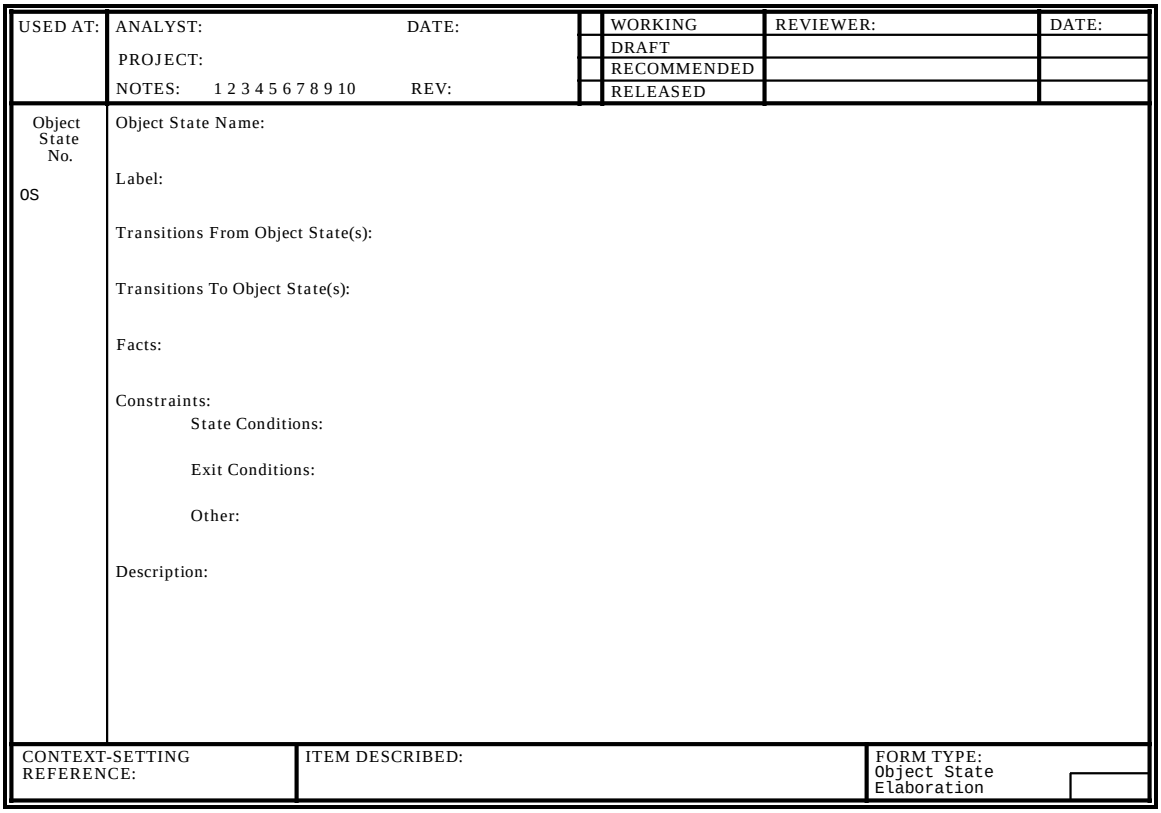

Figure 4-11
Object State Elaboration Form

The following list contains a description of the fields that appear on an object state elaboration form:

1. **Object State No.:** A unique identification number for the object state, prefaced by the letters «OS.»

2. **Object State Name:** The name of the object state.

4. **Label:** The Object Schematic name, some part of the name, or an abbreviation used for convenience when displayed in an IDEF3 graphical element (e.g., when displayed in a referent).

5. **Transitions From Object State(s):** The object state(s) from which the object transitions.

6. **Transitions To Object State(s):** The object state(s) to which the object transitions.

7. **Facts:** Facts that hold about objects in this state.

8. **Constraints:** Constraints on objects in this state. In particular, three types of constraints are listed:

    a. *State Conditions* – Conditions that are individually necessary for an object to be in the state in question.

    b. *Exit Conditions* – Sufficient conditions for an object to no longer be in the state in question.

    c. *Other* – Additional constraints of interest.

9. **Description:** A textual description, or glossary, associated with the Object State. Any descriptive information that does not logically fit into any of the other fields in the document may be placed here.

The transition link elaboration document is used to capture the elaborations of the transition links in an Object Schematic. A transition link elaboration document is constructed for every transition link represented in the Object Schematic. A transition link itself only indicates what object states can transition to which others. Hence, its elaboration consists only of the transition conditions for instances of its source state in an attempt to begin a transition that brings about an instance of its destination state and of the entry conditions that objects arising from its source state must meet to enter the destination state. In addition to containing this information, the transition link elaboration document also contains a unique transition link number for the link as well as the name of the Object Schematic that contains it (in the context setting reference field). The transition link elaboration document is shown in Figure 4-12.

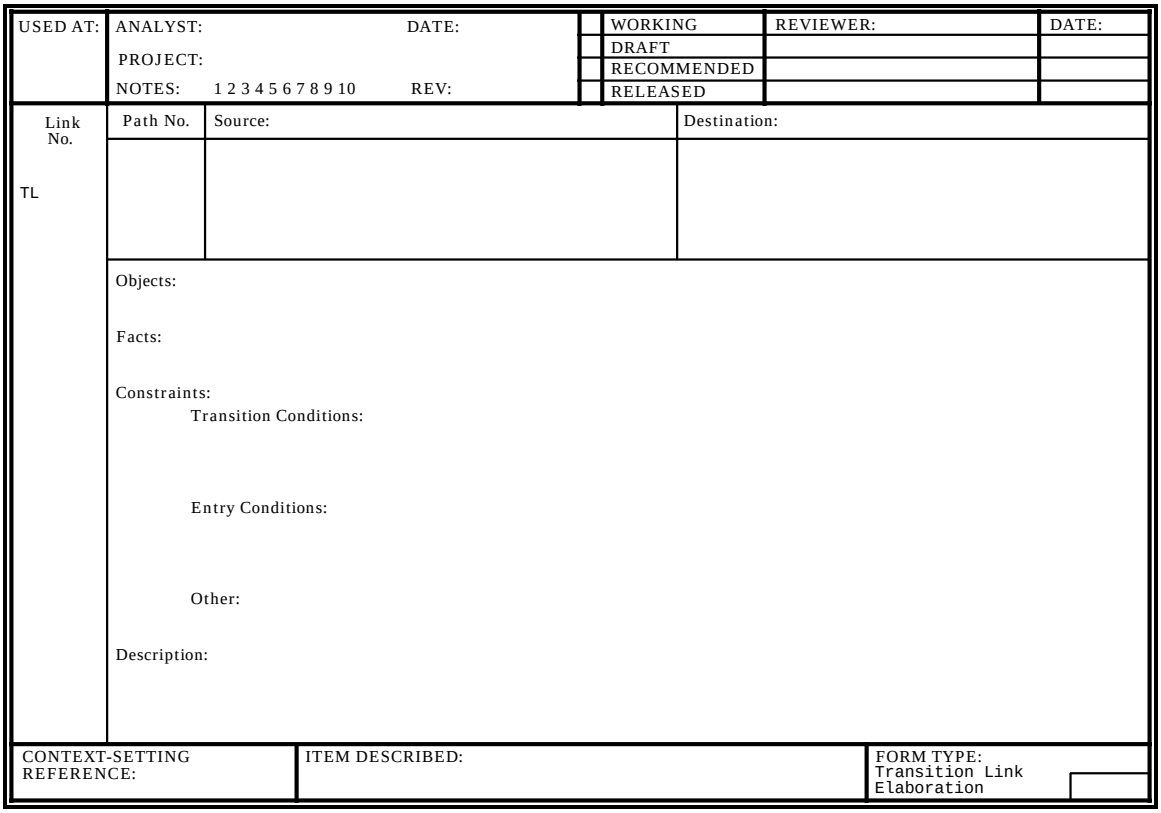

Figure 4-12
Transition Link Elaboration Form

The fields that appear on the transition link elaboration form include:

1. **Link No.:** A unique identification number for the transition link, prefaced by the letters «TL.»

2. **Path No.:** A link path number, comprised of the Link No. and a unique integer, separated by a period. For example, given a transition link TL1 that separates into three alternative paths following a junction, the path link numbers would be TL1.1, TL1.2, and TL1.3.

3. **Source:** Name of the source IDEF3 element (e.g., Object State) indicated by a link.

4. **Destination:** Name of the destination IDEF3 element (e.g., Object State) of the relation indicated by the link.

5. **Objects:** All significant objects (types or instances) that participate in the relation represented by the transition link.

6. **Facts:** Noteworthy, nonconstraining facts involving the objects that participate in the relationship represented by the transition link.

7. **Constraints:** Constraints on objects in this state. In particular, three types of constraints are listed:

    a. *Transition Conditions* – Conditions that are individually necessary and jointly sufficient for there to be an attempted transition from the source to the destination.

    b. *Entry Conditions* – Sufficient conditions for an object to enter the state given an object (possibly different) in the source state of that link that has met the relevant transition conditions.

    c. *Other* – Additional constraints of interest.

8. **Description:** The glossary associated with the transition link. Any descriptive information that does not logically fit into any of the other fields in the document may be placed here.

As appropriate, objects, facts, and constraints uniquely associated with a particular link path will be identified.

At this point, it may be useful to identify other objects and relations that can provide additional context-setting information relevant to the transition. Two elaboration forms are provided to assist with this task: the *object elaboration* document and the relation *link elaboration* document.

The object elaboration document is used to further characterize context-setting objects included in the Transition Schematic which are not directly involved in the focus transition. An example form for the object elaboration document is provided in Figure 4-13.

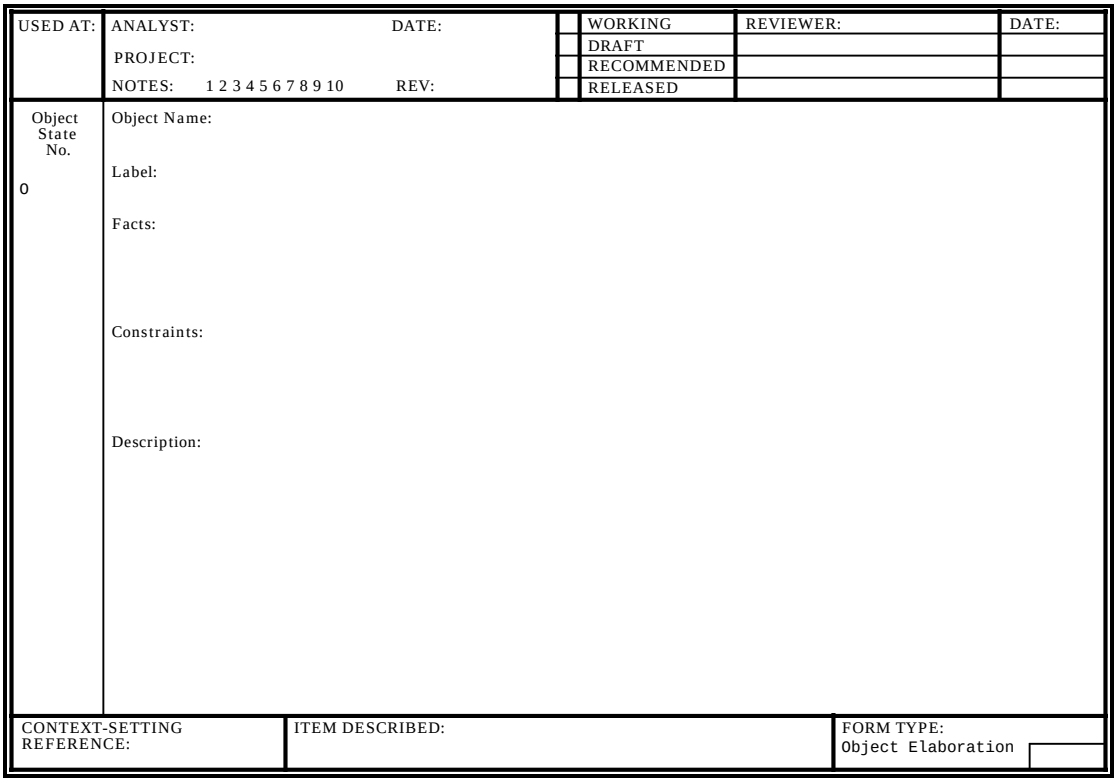

Figure 4-13
Object Elaboration Form

The following list describes the contents of each field on the object elaboration document.

1. **Object State No.:** An object number, prefaced by the letter «O» (for Object), that uniquely identifies the Object.

2. **Object Name:** This section contains the Object Name.

3. **Label:** This section contains the Object Label (i.e., the Object Name, some part of the Name, or an abbreviation).

4. **Facts:** This field lists facts about instances of the Object.

5. **Constraints:** This field lists constraints on the Object, i.e., facts about what must hold in all instances of the Object.

6. **Description:** This field contains a glossary entry (textual description) for the Object. Typically, the glossary entry provides a textual recount of the information already in the object, fact, and constraint lists.

The relation link elaboration document is used to further characterize the relations between objects and object states in an Object Schematic (other than the «transitions-to» relation). Figure 4-14 illustrates an example relation link elaboration form serving this purpose.

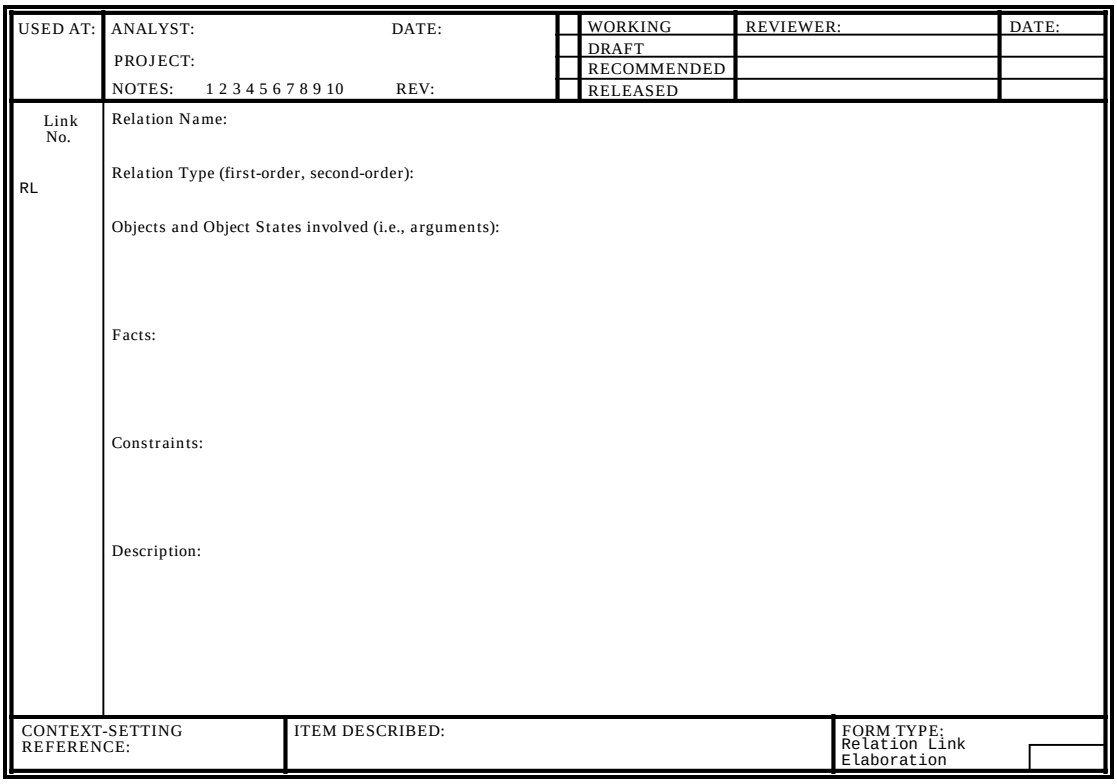

Figure 4-14
Relation Link Elaboration Form

The fields contained on the relation link elaboration form are as follows:

1. **Link No.:** A link number, prefaced by the letters «RL» (for relational link), that uniquely identifies the relation link within the description.

2. **Relation Name:** This section contains the Relation name.

3. **Relation Type:** A description of the number of places and order of the relation (e.g., 2-place, first-order).

4. **Objects and Object States involved (i.e., arguments):** A list of the Object(s) and Object State(s) involved in the relation.

5. **Facts:** Noteworthy nonconstraining facts involving the objects that participate in the relationship represented by the relation link.

6. **Constraints:** Noteworthy constraints that hold between the participating object(s) and object state(s) or between some of their constituent objects.

7. **Description:** The glossary associated with the relation link. Any descriptive information that does not logically fit into the other fields in the document may be placed here.

Incrementally Refine and Validate IDEF3 Process Descriptions

Motivation

The leverage of IDEF3 for description capture is particularly noticeable when validation activities are undertaken. Conventional process modeling often force users to gloss over gaps in the description or simplify facts with idealizations. IDEF3 does not impose such restrictions. It provides a flexible yet formal mechanism for recording the facts known about the operation of the system. Gaps and inconsistencies are made obvious in the schematic layouts specifically to bring them to the attention of analysts and domain experts. Likewise, capturing multiple viewpoints of a process serves to highlight differences. A better understanding of the process is achieved by both the experts and the analyst as they attempt to fill gaps and resolve inconsistencies both in a view and between views. This creates an understanding of how perceptions about the process differ between experts.

In contrast, conventional techniques typically present the analyst's assumptions about the process interspersed with his understanding of the expert's description. This model is then presented to domain experts for validation. Often, the expert, either in the interest of expediency or because of increasing pressure for consensus, signs off on a process model without completely understanding the implications. Using IDEF3, it is possible to use process description schematics as discussion focal points to resolve inconsistencies (if any) between differing viewpoints of how a process works.

Types of Validation

Validation is the process of checking and ensuring that the IDEF3 process description constructed is both syntactically and semantically correct. As one might assume from this definition, there are two types of validation: *syntactic* and *semantic*. Syntactic validation involves ensuring that the constructed IDEF3 schematic conforms to the grammatical rules of the IDEF3 language. Semantic validation involves ensuring that the statements made in the IDEF3 description accurately capture the assertions of the domain expert.

The IDEF3 method provides the user considerable freedom in terms of how these descriptions can be structured; the syntax of the language imposes few restrictions on possible schematic layouts. These restrictions, or rules ensure that the syntax and semantics of the constructed descriptions capture the user's intent. Moreover, these validation checks try to enforce standardization between the potential users of the language in a manner that enhances the utility of the method as an unambiguous means of communication.

Build and Distribute Kits

A primary means of validating IDEF3 process descriptions is through the review and approval of kits[13]. Kits represent portions of the total description that have reached some state of completion. The kit review task can be performed any time during the description development effort as a mechanism for acquiring additional facts or when a significant portion of analysis work has been completed (e.g., completion of the initial lists of UOBs and objects, completion of one or more Object Schematics, completion of a Process Schematic). Kit production and the associated review cycle (discussed below) provide a disciplined approach that results in an accurate description of the process.

Roles in the Kit Review Process

The team member roles described earlier in Section 4 are further specialized for the kit review process. The roles of the personnel involved in the kit review process are as follows:

1. **Analyst:** IDEF3 expert who is the primary developer of the IDEF3 description. The review process initiates and terminates with the analyst. The analyst relies on the domain expert for the technical content of the description, during both description capture and the kit review cycle.

2. **Reviewers:** All personnel involved in the review of IDEF3 kits.

3. **Commentors:** Reviewers who are knowledgeable in the application domain, and proficient enough in IDEF3 to offer structured comments in writing. Commentors read the material produced by analysts and verify its technical accuracy. They are responsible for finding errors and suggesting improvements in the IDEF3 process description. The commentor determines whether the purpose has been met, and whether errors or oversights exist. Commentors are authorized to make written suggestions during the review process.

---

[13] The genesis of this kit review procedure comes directly from the original IDEFØ Function Modeling «yellow book,» AFWAL-TR-81-4023. This was done to maintain consistency among the IDEF methods. The input from this document is greatly appreciated and acknowledged.

4. **Readers:** Reviewers to whom IDEF3 kits are distributed for informational purposes only. Readers are often individuals from whom analysts may have obtained information via interviews.

5. **Librarian:** A person assigned the responsibility of maintaining files of project-related documents and description artifacts, making copies, distributing IDEF3 kits, and keeping records.

A «role» is not related to an individual's job title; therefore, the same person may perform several roles.

The IDEF3 Kit Review Cycle

Kits represent portions of an IDEF3 process description that have reached some state of completion. These draft portions of a description are distributed for review in the form of a standard IDEF3 kit. The IDEF3 kit review cycle illustrated in Figure 4-15 is based on the kit review process for other IDEF methods. For clarity, the following steps do not mention the librarian, but focus on the interaction between the analyst and commentor. With large systems, the role of the librarian is essential. In smaller efforts, that role may be assumed by the analyst.

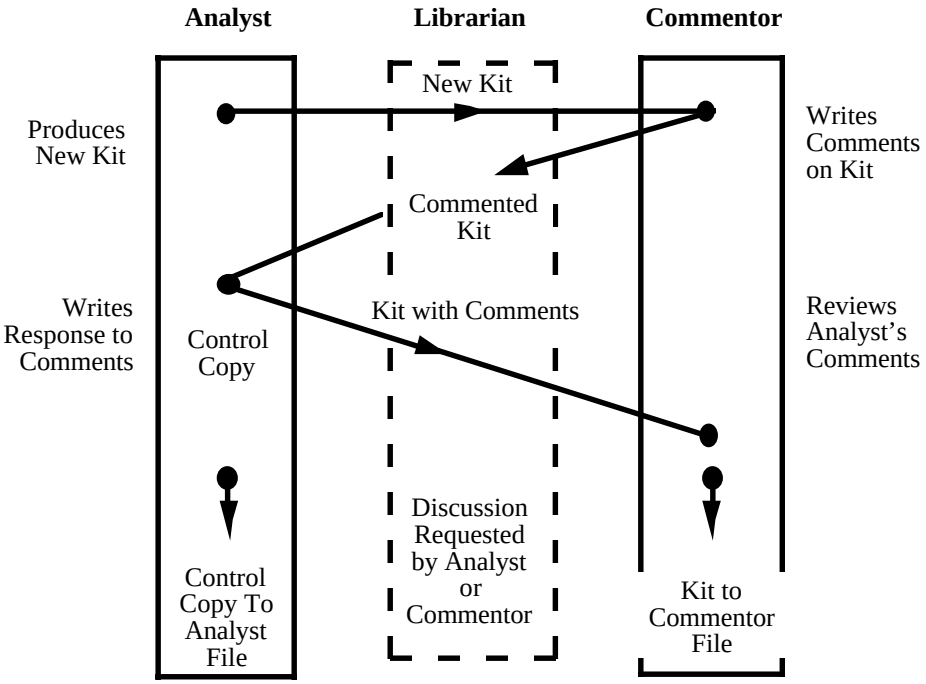

Figure 4-15
IDEF3 Kit Cycle

The following are the major steps in the IDEF3 kit review cycle.

1. The analyst assembles a kit (e.g., a pool kit, a scenario kit, an object kit, or a description kit with Process Schematics and Object Schematics). The analyst retains one copy and gives one copy to the commentor for review.

2. The commentor studies the kit within an agreed time period. The main purpose of this review is to determine whether the description complies with the overall goals and context of the development effort. Comments are made directly on the *schematics*, other documents in the kits, and the cover sheet. The kit with comments should be returned to the analyst by the date indicated on the cover sheet.

3. The analyst responds to the comments directly on the commentor's copy of the kit. The analyst may agree with the comments, noting it on his working copy and incorporating it into the next version of the IDEF3 description. If there are disagreements, the analyst notes the points of disagreement on the kit and returns the kit to the commentor.

4. The commentor will read and file the returned kit if the analyst's responses are satisfactory. Otherwise, a meeting between the commentor and the analyst is arranged to resolve the disagreements.

5. This cycle continues until a mutually acceptable (to the analyst and commentor) IDEF3 description is produced.

Throughout the cycle, a project librarian handles copying, distribution, filing, and transfer of IDEF3 kits between the analyst and the commentor (see Figure 4-15).

The results of the IDEF3 kit cycle are an IDEF3 description to which the analyst and the commentor have contributed, and, if necessary, a list of issues that require management action. A valuable byproduct of this review cycle is a recorded history of the review process.

Types of IDEF3 Kits

IDEF3 kits have a structure similar to those for other IDEF methods. There are three types of IDEF3 kits:

1. **Scenario Kits** address one scenario and all or part of its associated documentation. The following items may appear in a scenario kit.

a. Process Schematics and all associated UOB decompositions. Some of the review kits created early in the development process may omit some of the decompositions.

b. All available UOB elaborations and link specifications. Some of the scenario kits created early in the development process may omit some or all of these.

2. **Object Kits** address one or more objects and the associated Object Schematics, their descriptions, and their associated object state descriptions.

3. **Description Kits** are created in the later stages of a development effort. A description kit is a compilation from the completed scenario and object kits for a given project. It contains all the scenarios in the IDEF3 description and their associated documentation. An approved description kit is one of the final deliverables in a development effort.

Scenario kits can provide any level of detail from a single-scenario Process Schematic to a complete process description that contains all elaborations and UOB decomposition schematics. Description kits can also provide any level of completion; however, they reflect the current status of the entire project as opposed to that of the single scenario of a Scenario Kit.

Guidelines for Analysts and Commentors

Commentor Guidelines

No set pattern of questions and rules can be adequate for commenting, since subject matter, style, and technique vary widely. However, guidelines exist for improving quality. The major criteria for quality are: Will the document communicate well to its intended audience? Does it accomplish its purpose? Is it factually correct and accurate, given the bounded context? The following are overall guidelines for commenting:

1. Make notes brief, thorough, and specific. As long as the analyst understands that niceties are dropped for conciseness, communication is easier and less cluttered.

2. Use the ⓝ notation to identify comments. To write a ⓝ -note, check the next number off the NOTES list, number the note, circle the number, and connect the note to the appropriate part with a squiggle «~.»

3. Make constructive criticisms. Try to suggest solutions rather than just making negative comments.

4. Take time to gather overall comments. These may be placed on the cover or a separate sheet. (Don't gather specific points on this sheet if they belong on the individual pages.) Agenda items for analyst/commentor meetings may be summarized. Make agenda references specific.

The time spent critiquing depends on several different factors: familiarity with the subject, the number of times something has been reviewed, the experience of the commentor and analyst, etc. An IDEF3 kit returned to an analyst with no comments means that the commentor is in total agreement with the analyst. The commentor should realize that there is a shared responsibility with the analyst for the quality of the work.

Analyst/Commentor Interchanges

When a commentor returns an IDEF3 kit, the analyst responds by putting a «√» or «X» by each ⓝ-note. A «√»means the analyst agrees with the commentor and will incorporate the comment into the next version of the IDEF3 kit. An «X» means the analyst disagrees and requires a reason to be noted where the comment appears. After the analyst has responded to all comments, the IDEF3 kit is returned to the commentor.

After reading the analyst's responses, the commentor identifies remaining points of disagreement and requests a meeting with the analyst. This specific list of issues forms the agenda for the meeting.

Meeting Rules

Until comments and reactions are on paper, commentors and analysts are discouraged from conversing.

When a meeting is required, the procedure is as follows.

1. Each meeting should be limited in length.

2. Each session must start with a specific agenda of topics to be considered; discussions must not deviate from these topics.

3. Each session should terminate when the participants agree that the level of productivity has dropped and individual efforts would be more rewarding.

4. Each session must end with an agreed list of action items which may include the scheduling of follow-up sessions with specified agendas.

5. In each session, a «scribe» should be designated to take minutes and note actions, decisions, and topics.

6. Serious, unresolved differences should be handled professionally (i.e., documenting both viewpoints).

The result of the meeting should be a written resolution of the issues or a list of issues to be settled by appropriate managerial decision. Resolution can take the form of more study by any participant.

Contents of IDEF3 Kits

An IDEF3 kit is a technical document. It may contain schematics, text, glossaries, decision summaries, elaborations, background information, or other relevant material packaged for review and comment.

General Guidelines for Kit Preparation

To avoid oversights, review the IDEF3 kit as if it were the only information available. Add points of clarification as brief notes on the IDEF3 kit. Glossary definitions for terms that appear in the IDEF3 kit should always be appended as support material.

Gather helpful materials and append these for the commentor's benefit. Never use this supplemental material to convey information which should properly be conveyed by the schematic itself. Whenever possible, use the most natural means of communication to show details that are important for the reader in understanding the concepts. Combine all material with a completed cover sheet and submit to the librarian.

The Kit Cover Sheet

The Kit Cover Sheet distinguishes the material assembled with it as an IDEF3 kit. The cover sheet has fields for analyst, date, project, document number, title, status, and notes. The following describes what information should be provided in the fields of an IDEF3 Kit Cover Sheet (see Figure 4-16).

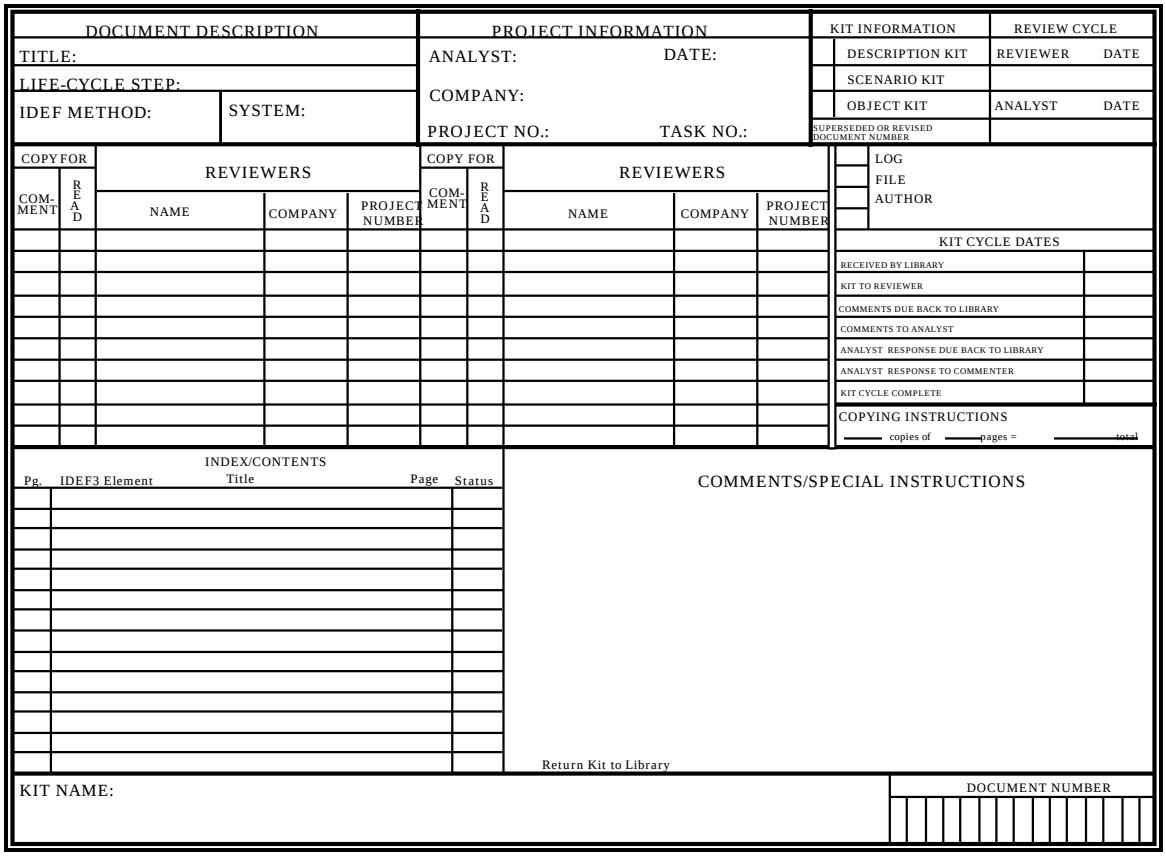

Figure 4-16
IDEF3 Kit Review Cover Sheet

The following sections and their contents are provided on the IDEF3 kit review cover sheet.

1. **IDEF3 Process Description/Document Description:**

   a. *Title* – Should be descriptive of the IDEF3 kit.

   b. *Life-Cycle Step* – «AS-IS» or «TO-BE» (does the kit contain a description of something that is or something that might be).

   c. *System* – Acronym for System or Subsystem.

2. **Project Information:**

   a. *Analyst* – Name of person submitting the IDEF3 kit.

   b. *Date* – Date sent to library.

   c. *Company* – Name of the company submitting the IDEF3 kit.

3. **IDEF3 Kit Information:** Check Description Kit, Scenario Kit, or Object Kit. Indicate document number assigned by the librarian.

4. **Review Cycle:** To be signed and dated after review by commentor and analyst.

5. **Index/Contents:** List the Scenario, Decomposition, Object, and Object State (if relevant) names along with the page number where they can be found in the document. An additional sheet called the IDEF3 Kit Contents Sheet (see Figure 4-17) is also filled out if necessary along with the Kit Cover Sheet.

6. **Comments/Special Instructions:** Any other information for the reviewers. This can also be used for special instructions to the librarian about handling the document. The library also uses this field for special instructions to the recipients of IDEF3 kits.

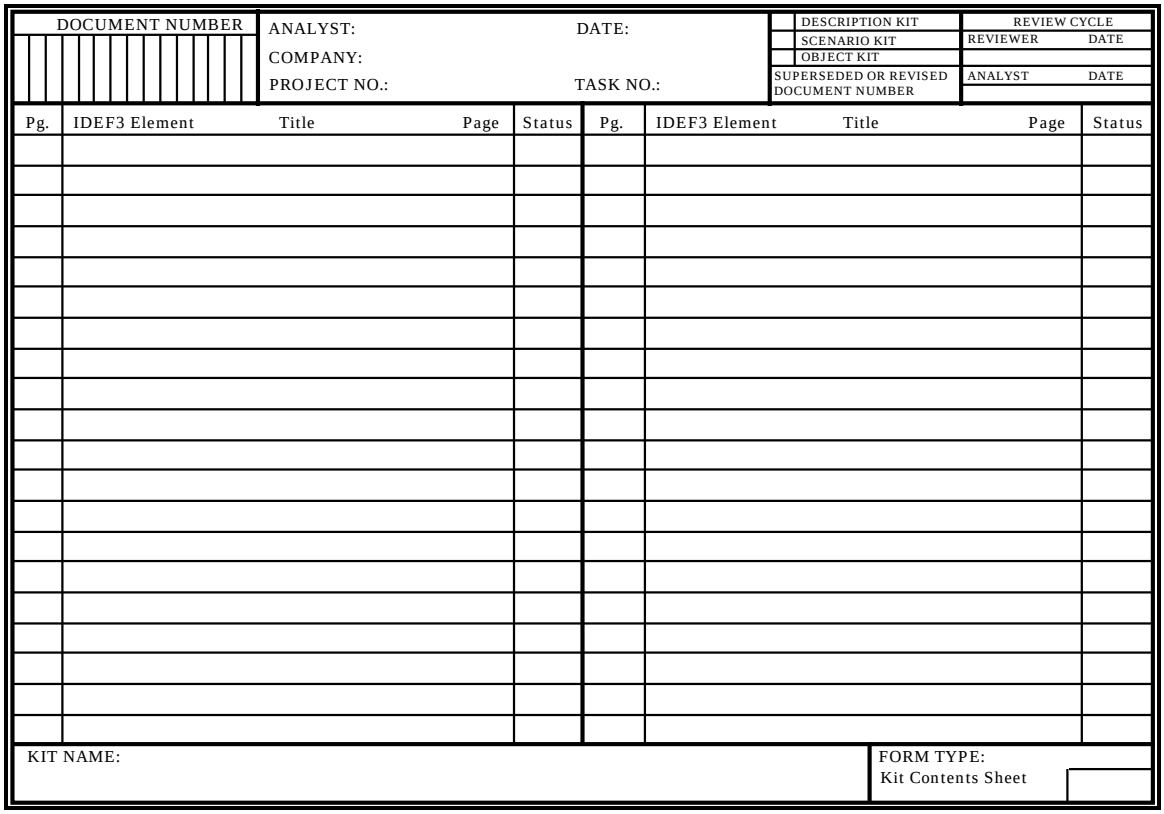

Figure 4-17
IDEF3 Kit Contents Sheet

The Kit Schematic Form

The Kit Schematic Form, as shown in Figure 4-18, has minimum structure and constraints. The sheet supports only the functions important to the discipline of structured analysis: (1) establishing a viewpoint, (2) cross-referencing between sheets of paper, and (3) documenting notes about the contents of each sheet. The form is divided into three major sections: (1) *Working Information* (top), (2) *Message Field* (center), and (3) *Identification Fields* (bottom).

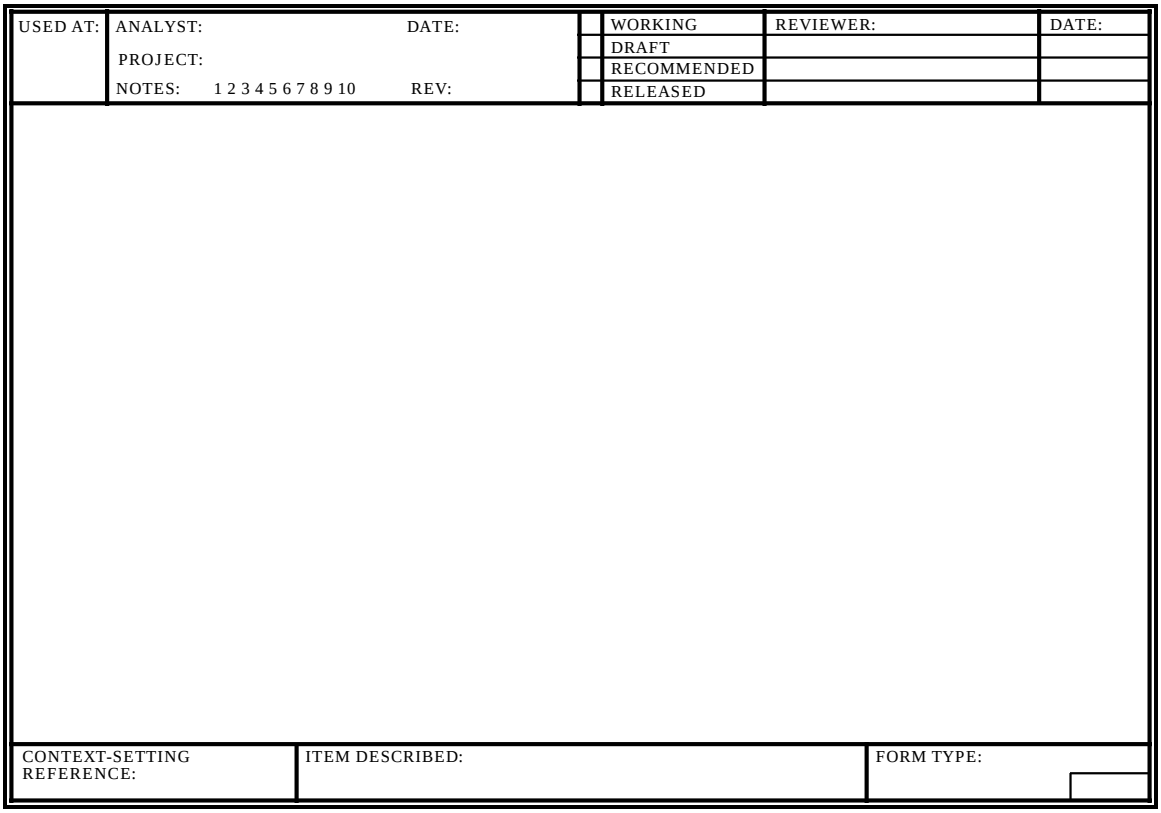

Figure 4-18
IDEF3 Kit Schematic Form

The form is designed so that the working information at the top of the form may be cut off when a final «approved for publication» version is completed. The schematic form should be used whenever printed documents are used.

The following are the subfields of the **Working Information** field.

1. **Used At:** This is a list of schematics, other than the immediate context, which use this sheet in some way.

2. **Analyst/Date/Project:** This documents who originally created the schematic, the date it was first drawn, and the project title under which it was created. The «date» entry may contain additional dates, written below the original date. These dates represent *revisions* to the original sheet. If a sheet is re-released without any change, no revision date is added.

3. **Notes:** This provides a check-off for ⓝ notes written on the schematic sheet. As comments are made on a page, the notes are successively crossed out. This

provides a quick check for the number of comments, while the circled number provides a unique reference to the specific comment.

4. **Status:** Four status classifications provide a ranking of approval: working, draft, recommended, and released.

    a. *Working* – The schematic is a major change, regardless of the previous status. New schematics are, of course, working copy.

    b. *Draft* – The schematic is a minor change from the previous schematic and has reached some agreed-upon level of acceptance by a set of readers. Draft schematics are those proposed by a project leader, but not yet accepted by the project team.

    c. *Recommended* – Both this schematic and its supporting text have been reviewed and approved by the project team. This schematic is not expected to change.

    d. *Released* – This page may be forwarded *as is* for final release or publication.

5. **Reader/Date:** This area is for the commentor to initial and date each form.

The **Message Field** contains the primary message to be conveyed. The field is normally used for schematics, but the field can be used for any purpose (e.g., glossary, checklists, notes, sketches).

The **Identification Fields** are as follows.

1. **Context-Setting Reference:** The information provided in this field helps to establish a context for interpreting the information in the message field. That context is established with a reference identifier that is the unique IDEF3 element number (e.g., Scenario 1, Decomp 1.1, UOB43, PL31, J5, O6, OS22). Use of the term «Global» for the reference identifier may be used to indicate a global context. For example, if a scenario elaboration were displayed in the message field, the context-setting reference would be «global.» If, on the other hand, a Process Schematic were displayed in the message field, the context-setting reference would indicate a scenario number or a parent UOB number.

2. **Item Described:** This field contains the name of the material presented in the message field of the schematic form. If the message field contains a schematic, the contents of the title field must precisely match the name written in the parent box.

3. **Form Type:** The standard IDEF3 kit form may be used as the basic structure for all forms other than the kit cover sheet used during description development. This field is used to establish how the standard IDEF3 kit form is being used. Recommended form types include the Description Summary form, the Kit Contents Sheet, the Process and Object Schematic Summary forms, the Source Material Log, the Source Material Description form, individual pool forms, and IDEF3 element elaboration forms.

Review Progress and Make Adjustments

Throughout the description capture effort, the project team will find it necessary to frequently review the purpose and scope of the project and assess progress. Adjustments requiring some redefinition of scope often surface in projects with a purpose aimed at solving some ill-understood problem situation. Frequent assessments of progress toward satisfying the purpose of the project promote early detection of high payoff opportunities, and limit the time and expense used in less fruitful activity. These discoveries often motivate subtle or dramatic changes in scope. When the need for such changes arise, the client should be notified and his or her approval should be sought. Analysts may also recognize the need to augment the use of IDEF3 with other methods.

Using other IDEF Methods in Process Description Capture

IDEF3 was designed to work independently or in concert with other IDEF methods. Methods and techniques outside the IDEF family have been successfully applied with IDEF3 on a number of projects.[14] In these cases, it is necessary to establish clear roles for each method. It is also important to clearly define the conventions that will be used in applying each method. Selecting the appropriate set of methods for a given project depends on the nature and form of the available information and on the purpose of the project. Each IDEF method is tailored for a unique set of information and cognitive support applications. For example, the IDEFØ Function Modeling and IDEF1 Information Modeling methods are useful for analyzing complex situations. The IDEF5 Ontology Description Capture method provides additional expressive power for describing object structures and relations. The choice to apply more than one method over the course of a project underscores the nature of methods as mechanisms designed to support predominantly narrowly-scoped tasks that may be applicable across a wide range of general and project-specific systems engineering frameworks. These frameworks serve to establish a context for the required integration among the multiple tasks, and consequently among the methods supporting those tasks.

[14] See, for example, papers describing the use of IDEF3 in the Proceedings of the IDEF Users Group.

Using IDEFØ with IDEF3

IDEFØ model and the UOBs in an IDEF3 Process Description, IDEF3 is not intended to be a replacement for IDEFØ. If the system being analyzed is very large (e.g., Manufacture Aerospace Product), precedence relations may not be evident. In these cases, it is often better to start with an IDEFØ model. Such a model can then be decomposed to a level where the precedence relations among activities become prominent. On the other hand, if the facts collected can be organized into a cohesive story, it is generally better to formulate the IDEF3 process description first, then abstract an IDEFØ model from that description. The IDEF3 method was designed with this interaction in mind.

The IDEF3 syntax recognizes this relationship by providing a means of referencing associated IDEFØ activities from within the IDEF3 UOB. All UOB boxes have a field (see lower right of Figure 4-19) for providing a reference to an activity in an IDEFØ model, or comparable function or process model (e.g., a node in a Logical Data Flow Diagram or HIPO chart).

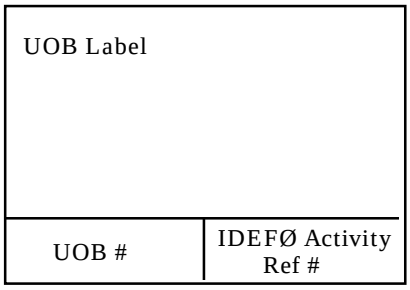

Figure 4-19
Unit of Behavior Fields

The reference scheme in IDEF3 assumes that zero, one, or many IDEFØ activities will map onto a single UOB. In cases where the UOB maps to only part of an IDEFØ activity, the activity referent should point to the set of child activities in the IDEFØ model that is actually involved. If the IDEFØ model is not defined to a low enough level of detail, the extent of the mapping should be described in the UOB elaboration. As UOBs are identified, IDEFØ references should be included.

Using IDEF1 and IDEF1X with IDEF3

In many large IDEF3 development projects, IDEF1 and/or IDEF1X models are available prior to the project initiation. These can help identify the objects for Object Schematics. The entity class number or attribute class (in IDEF1), or the entity number or attribute (in IDEF1X) that relates to each object or object state, should be referenced in

the glossary of the Object Schematic or the appropriate Object Schematic Description form.

While capturing process descriptions is generally straightforward, determining the business rules that are supported by the information system is more difficult. There are often hidden pockets of information that constitute the «informal» information system of the enterprise. One of the primary roles of information modeling is to define the informal and formal information system. IDEF3 descriptions can be developed either before or after the development of information models. When developed prior to information modeling, IDEF3 process descriptions can help users develop information models by focusing domain expert attention on the information required to support their process. The resulting information models constitute a process-centered view of the information requirements of the enterprise. Integrating a number of process-view information models will eventually yield a comprehensive information model that can be used to develop enterprise-wide data standards.

Using IDEF5 with IDEF3

Because of the importance process kinds can have in the definition of a domain ontology, IDEF5 permits one to refer to them as no less than object kinds. However, there are two distinct contexts in which such references can occur, and the information that is kept about a process kind will differ depending on the context. If a process kind **P** is referred to in the description of a transformation or transition involving two kinds of objects, then the «internal» character of **P** is described in accordance with the IDEF3 process description capture method. That is, **P** is described in terms of the object kinds it involves, their properties, and the relevant relations that hold between instances of those kinds when the process in question is instantiated. In particular, in such contexts, the usual sort of information kept about an object kind—its defining properties and so forth—is not kept about the process kind.

On the other hand, it may be important for understanding a domain not only to know how objects are involved in the internal structure of a process, but also—as with object kinds generally—how one kind of process relates logically to another kind of process, independent (in general) of the details of its internal structure. For instance, **manufacturing process planning** is a subkind of **planning**. In these cases, process kinds are characterized using the procedure and language constructs provided within the IDEF5 ontology capture method.

IDEF3 DEVELOPMENT: MATERIAL ORDERING PROCESS EXAMPLE

The example description of a company's purchase order process presented in this section demonstrates the use of the IDEF3 method in a common setting. The example includes some tips to help the user avoid common errors. Moreover, justifications for applying a structured process to description development are documented as a guide to the novice user.

Define Purpose and Context

Assume that an owner of a business is interested in using IDEF3 to document the material ordering process to assist with new worker training and enforce company purchasing standards. Thus, he hires an analyst to develop an IDEF3 Process Description for the scenario *Order Material*. Assume that this project stems from the owner's desire to record how purchase requests are processed for the benefit of new employees. One advantage of applying IDEF3 in this situation will be that a new employee can quickly understand how to acquire needed material by referring to the IDEF3 Process Description, without forcing the owner to spend time communicating this knowledge. In this example, the boundaries of the problem will be restricted to activities within the company. Only the information needed to clearly specify the workings of the purchase order process to a new employee will be captured. This purpose and context would be entered on the IDEF3 Process Description summary form. At this stage of the description development process, the analyst would normally identify candidate scenarios and begin an IDEF3 scenario pool. The contents of this pool will be refined and maintained throughout the life of the project.

In this example, only three IDEF3 project team roles are illustrated: (1) the analyst (the IDEF3 expert), (2) the domain expert (the business owner), and (3) the client (also the business owner). The domain expert and the client are usually not the same individual. The remainder of this section will often refer to these individuals by their project role names.

Collect Data

Having defined the project, the analyst prepares for and conduct data collection activities. One of the most valuable mechanisms for this data collection is the interview. For this example, we will focus primarily on this aspect of data collection.

Interview Domain Expert and Acquire Initial Description

Recognizing that well-planned and well-executed interviews are critical, the analyst prepares carefully. When the scheduled interview time arrives, the analyst might begin by asking, «How does one go about purchasing material once the need has been identified?» Suppose the domain expert answers with the following description:

> «The first thing we do is request material using a purchase request form. Then the Purchasing Department either identifies our current supplier for the kind of material requested or sets out to identify potential suppliers. We like to develop long-term relationships with our suppliers. That means we will always use a current supplier whenever possible. In return, we expect their highest quality products on time and at reasonable prices. Right now, we have contracts or informal agreements with 7 trusted suppliers. If we have no current supplier for the needed item, Purchasing requests bids from potential suppliers and evaluates their bids to determine the best value. Once a supplier is chosen, Purchasing orders the requested material.»

During the interview, analysts often find it helpful to request to see copies of source material associated with the process. The analyst may also find it helpful to obtain copies of relevant segments of policy and procedure manuals, previously developed models, purchase requests, supplier lists, solicitations, bid evaluation reports, purchase orders, and so forth. During the example interview, let us assume that the domain expert provides a copy of the purchase request form. The analyst notices three signature blocks at the bottom of the page—one titled «Requester,» another titled «Account Manager,» and a third titled «Purchase Authorization.» This leads to a new line of questioning which the domain expert answers as follows.

> «To request material we must first prepare a purchase request. The information required on the purchase request form includes the item description, the number of items needed, the required receipt date (if applicable), the number of the account that will fund the purchase, a written justification for the stated need, and the requester's printed name and signature. The requester must then obtain the account manager's approval, or that of the designated backup, for the purchase. Account managers, or their designated backups, are responsible for, and must approve, all purchases made against their project accounts. After the account manager approves the purchase, an authorization signature may be required. To avoid a potential conflict of interest, the person initiating the purchase request cannot be the same individual as the one who approves or authorizes the request. Once all the appropriate signatures have been obtained, the requester submits the signed purchase request to Purchasing.

Purchasing then orders the requested material. The purchase request is then tracked as an issued purchase order.»

Note that the second description, although more detailed, omits any description of how the supplier is identified, although this information was deemed important enough by the domain expert to include it in the first description. In practice, the completeness of the description provided by an interview will depend upon several factors:

1. The amount of time the domain expert is willing (or allowed) to devote to the interview.

2. The experience and domain-specific knowledge of the interviewer.

3. The domain expert's knowledge of the process being described.

During the interview with the domain expert, the analyst will acquire the initial description that may include written documentation about the process. The purpose of acquiring a description is to represent how the system *actually* works, rather than how the domain expert *thinks* the system works (or how the domain expert thinks the system *should* work). Therefore, the analyst needs to correlate facts captured in the interview process with first-hand observations of the process. The analyst also must avoid completing the description with his or her own (often preconceived) knowledge about how the system ought to work. Thus, it is important that both the analyst and the domain expert understand that descriptions are often partial in nature and curb their desire to make them ideally complete.

Analyze Description for Data Identification

Once the interview is over, the analyst needs to carefully study the recorded notes and observations. This analysis identifies the objects, activities, facts, and constraints that occur in the description. This step is a list-making process.

When describing processes, individuals often focus on the key objects in the process and their roles in the process before actually describing the events or activities that occur during the process. The following is a list of objects that were identified in the description.

Material	Purchasing Department	Contract
Order	Potential Supplier	Bid
Current Supplier	Requester	Purchase Request
Account Manager	Backup	Project Account
Purchase Order	Chosen Supplier	

It is important that the analyst explicitly record the list of objects in the IDEF3 object pool for the following reasons.

1. The analyst may omit some of the objects at a later stage in the description capture process.

2. This list of objects from the first analysis often contains the primary objects in the process. Primary objects are those objects important enough to warrant the creation of an Object Schematic.

After identifying objects, the interview notes are examined to determine the activities/processes that occur in material ordering. The important activities are candidates to be represented as UOBs (activities, actions, or processes) in the description. However, at this stage of development, the sequence of the activities is not important. The primary goal is to list the candidate UOBs (as shown in the following list). These candidate UOBs would be listed in the IDEF3 UOB pool. It is likely that the list of UOBs is incomplete; however, this is not a matter of much concern at this stage. The first description yields the following UOBs.

1. Request material.

2. Identify potential suppliers.

3. Identify current supplier.

4. Request bids.

5. Evaluate bids.

6. Order requested material.

The second description yields four additional UOBs.

7. Prepare purchase request.

8. Obtain account manager's approval.

9. Obtain authorization signature.

10. Submit signed purchase request.

The final step in the interview analysis involves identifying and listing facts and identifying the constraints relevant to the processes described by the domain expert. *Facts* are assertions made about the objects. *Constraints* are distinguished conditions that are known to hold between the objects within a process, or between the processes themselves. To identify the occurrence of constraints, look for *negative* terms such as *not*, *never*, or *no* (as well as quantifiers like *every, all,* and *only*) in the recorded verbal description. The list of facts and constraints is likely to be incomplete early in the development. Further interviews or conversations with the domain expert will aid in making the lists of facts and constraints more complete. An initial list might include the following:

1. There are 7 current suppliers.

2. No one besides the designated account manager or his or her backup is allowed to approve purchases against their assigned account.

3. The requester cannot be the same individual as the one who approves or authorizes the request.

Formulate Process Schematics

Once the initial task of identifying objects, activities, facts, and constraints nears completion, the IDEF3 Process Schematic (or a set of schematics) is ready to be formulated. The observations recorded in the interviews are used as the basis for developing the Process Schematics. Candidate UOBs listed in the data analysis phase will be used in this step to construct the UOBs. Facts and constraints identified from the interview notes will be used to construct UOB elaborations. Developing a Process Schematic occurs in two major stages, (1) constructing UOBs in correct sequence and (2) developing UOB elaborations.

Layout Initial Process Schematic

The process of identifying the UOBs and specifying the precedence between them occurs in several steps.

> **Step 1.** Identify the left-most UOB in the process description, the UOB *Request Material*.
>
> **Step 2.** Identify the next UOB. In this example, two UOBs are possible: *Identify Current Supplier* or *Identify Potential Suppliers.*

The second step implies a split in the process flow, indicating the need to use a fan-out junction to represent the diverging flow. The analyst must determine the junction type that initiates the split. In this example, the Purchasing Department can perform only one of the two alternative activities; therefore, an XOR junction is used. The analyst may find it useful at this stage to create the partial schematic shown in Figure 5-1.

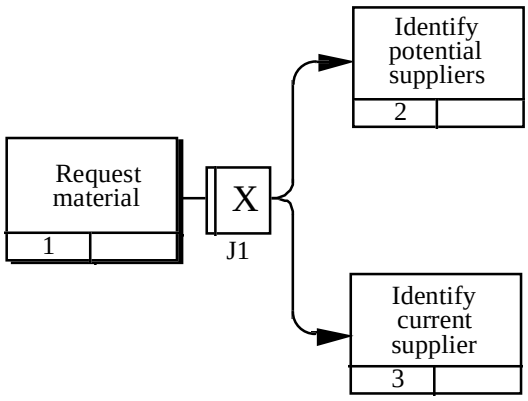

Figure 5-1
First Steps in Process Schematic Development

If a split in the process had not occurred, the development would have continued with the sequential drawing of UOB boxes until a split did occur. After a split, each process path is developed separately. These process paths may or may not converge within the context of the given description. The order in which the process paths are developed is a matter of preference.

> **Step 3.** The next step is to develop the path that begins with UOB 2. This path continues sequentially with the UOBs *Request Bids, Evaluate Bids,* and *Order Requested Material*. These UOBs result in the partial schematic shown in Figure 5-2.

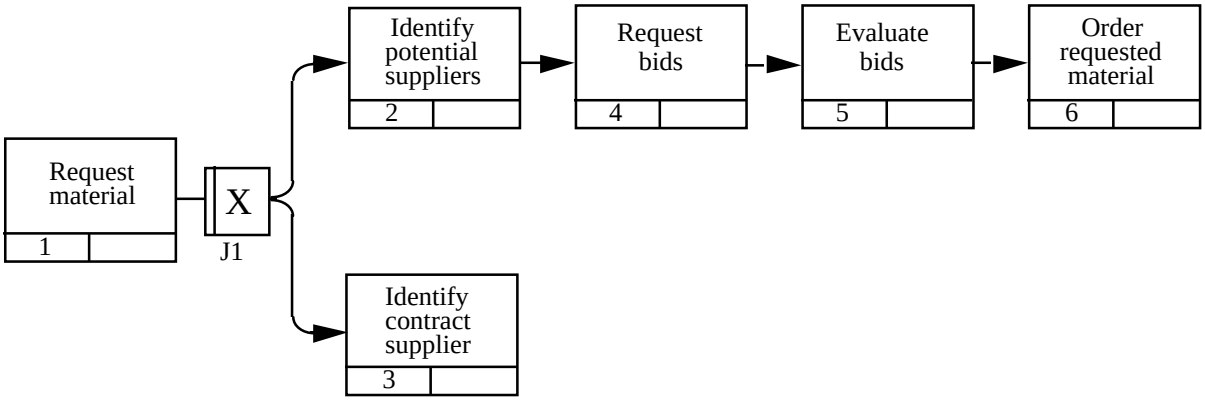

Figure 5-2
Schematic with the First Path Complete

Step 4. The fourth step is to complete the remaining path in Figure 5-2, resulting in the Process Schematic shown in Figure 5-3. Note that the UOBs retain the numbers assigned as they were placed in the activities list. The second path also results in the placement of an order for the requested material. This implies a convergence in the process flow and the need for a fan-in junction to represent the convergence. The analyst must determine the appropriate junction type for the convergence. In this example, only one of the two paths was possible, as indicated by the fan-out XOR that precedes each path. Therefore, a fan-in XOR junction is used.

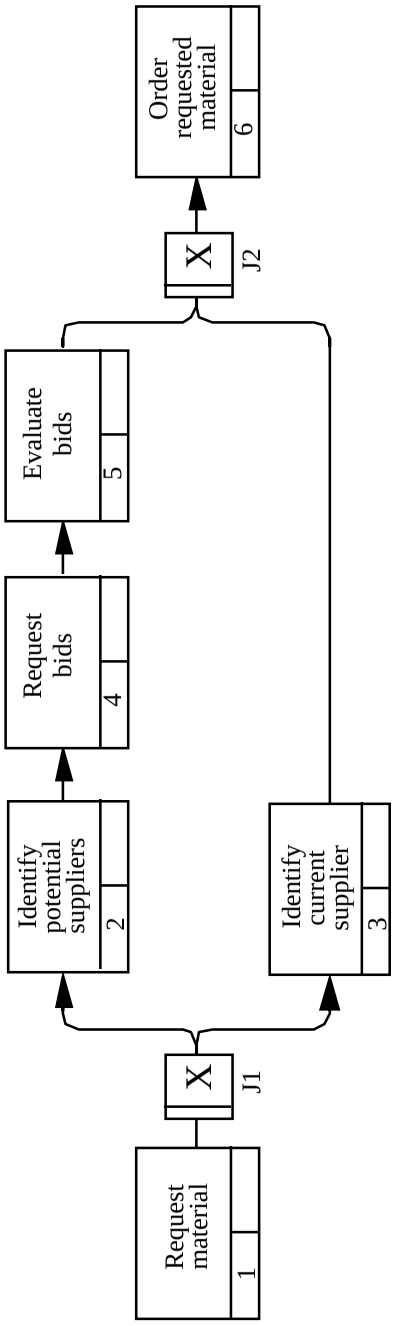

Figure 5-3
Schematic Near Completion

Step 5. When the schematic illustrated in Figure 5-3 is finished, there are still four activities in the list of potential UOBs. UOBs 7 through 10 are the domain expert's description of the *Request Material* UOB. Some analysts find it easier to begin schematic development at a more detailed level and later create decompositions to simplify the schematic. Others find it more convenient to begin schematic development at a higher level of abstraction and begin by only classifying the activities and viewpoints that they might want to investigate later through the creation of decompositions. Developing decompositions helps to keep the schematic simple and also affords the analyst additional opportunity to collect and organize alternative descriptions of how the *Request Material* activity is performed. This choice yields the schematic displayed in Figure 5-4.

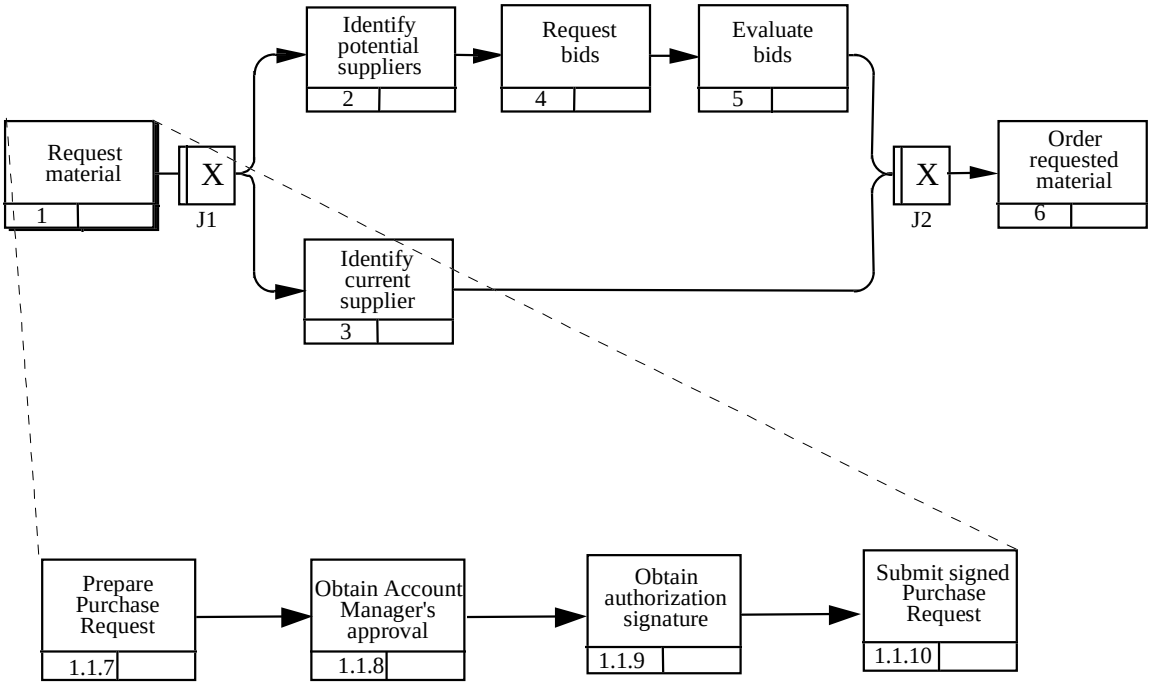

Figure 5-4
Complete Process Description Schematic Before First Review

Develop Elaborations

After the initial Process Schematic has been completed, elaborations must be added to each UOB as shown in Figures 5-5 through 5-9. In the initial attempt, these may be somewhat incomplete. One reason for this may be that the primary focus of the analyst in the first interview is on the objects and activities. This is particularly true in the development of either a description for a process with which the analyst was unfamiliar or a description of a large, complex process.

When the analyst is familiar with the process type, more information can be obtained about the particular process in the first interview. The analyst's questions would reflect this familiarity and in the first interview the analyst could determine how the process differs from other systems of this type. In developing the elaborations, the analyst needs to avoid allowing personal knowledge of the system type to influence the information placed in the elaborations.

The order in which the elaborations are developed is not important. It may often be useful to develop elaborations in parallel with developing the Process Schematic because, in some situations, this may aid the analyst in structuring the schematics. However, for this example, the initial elaborations were developed after the rest of the Process Schematic was complete. The elaborations that resulted are shown in Figures 5-5 through 5-9. For brevity in this example, we have not included the constraint lists in these elaborations. Recall that each link in the Process Schematic would generate a constraint entry in the elaborations of each linked UOB.

Review Process Schematic with Domain Experts

USED AT:	ANALYST: I. M. Modeler	DATE: 08 Feb 1995	X	WORKING	REVIEWER:	DATE:
	PROJECT: Process Description Capture			DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10	REV:		RECOMMENDED		
				RELEASED		

UOB No.		
UOB1	UOB Name: Request Material	UOB Label: Request Material
	Objects: Material Requestor Purchase Request	
	Facts:	
	Constraints:	
	Description:	

UOB No.		
UOB2	UOB Name: Identify potential suppliers	UOB Label: Identify potential suppliers
	Objects: Purchasing department Purchase Request	
	Facts:	
	Constraints:	
	Description:	

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE:
Scenario 1	Request Material UOB, Identify potential suppliers UOB	UOB Elaboration

Figure 5-5
Elaborations for UOBs 1 and 2

USED AT:	ANALYST: I. M. Modeler DATE: 08 Feb 1995	X	WORKING	REVIEWER:	DATE:
	PROJECT: Process Description Capture		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10 REV:		RECOMMENDED		
			RELEASED		

UOB No.		
UOB3	UOB Name:	Identify current supplier — UOB Label: Identify current supplier
	Objects:	Purchasing department; Material; Current supplier
	Facts:	7 current suppliers.
	Constraints:	
	Description:	

UOB No.		
UOB4	UOB Name:	Request bids — UOB Label: Request bids
	Objects:	Purchasing department; Potential supplier
	Facts:	
	Constraints:	
	Description:	

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE:
Scenario 1	Identify current supplier UOB, Request bids UOB	UOB Elaboration

Figure 5-6
Elaborations for UOBs 3 and 4

USED AT:	ANALYST: I. M. Modeler DATE: 08 Feb 1995	X	WORKING	REVIEWER:	DATE:
	PROJECT: Process Description Capture		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10 REV:		RECOMMENDED		
			RELEASED		

UOB No.		
UOB5	UOB Name:	Evaluate bids UOB Label: Evaluate bids
	Objects:	Purchasing department Bid Potential supplier
	Facts:	
	Constraints:	
	Description:	

UOB No.		
UOB6	UOB Name:	Order requested material UOB Label: Order requested material
	Objects:	Accounting and Finance department Purchase Request Current supplier Purchasing department Purchase Order Chosen supplier
	Facts:	
	Constraints:	Company will always use a current supplier when at all possible.
	Description:	

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE:
Scenario 1	Identify current supplier UOB, Request bids UOB	UOB Elaboration

Figure 5-7
Elaborations for UOBs 5 and 6

USED AT:	ANALYST: I. M. Modeler DATE: 08 Feb 1995		X	WORKING	REVIEWER:	DATE:
	PROJECT: Process Description Capture			DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10 REV:			RECOMMENDED		
				RELEASED		

UOB No.		
UOB7	UOB Name: Prepare Purchase Request	UOB Label: Prepare Purchase Request
	Objects: Requester Purchase Request Project account	
	Facts:	
	Constraints:	
	Description:	

UOB No.		
UOB8	UOB Name: Obtain Account Manager's approval	UOB Label: Obtain Account Manager's approval
	Objects: Requester Purchase Request Project account Account Manager Backup	
	Facts:	
	Constraints: No one besides the designated Account Manager or his backup is allowed to approve purchases against their assigned account.	
	Description: Requester cannot be the same individual as the one who approves the request.	

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE:
Scenario 1	Identify current supplier UOB, Request bids UOB	UOB Elaboration

Figure 5-8
Elaborations for UOBs 7 and 8

USED AT:	ANALYST: I. M. Modeler DATE: 08 Feb 1995 PROJECT: Process Description Capture NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

UOB No.	
UOB9	UOB Name: Obtain Authorization Signature UOB Label: Obtain Authorization Signature Objects: Requester, Project account, Purchase Request Facts: Constraints: Requester cannot be the same individual as the one who authorizes the request. Description:
UOB10	UOB Name: Submit signed Purchase Request UOB Label: Submit signed Purchase Request Objects: Requester, Purchasing department, Purchase Request Facts: Constraints: Description:

CONTEXT-SETTING REFERENCE: Scenario 1	ITEM DESCRIBED: Identify current supplier UOB, Request bids UOB	FORM TYPE: UOB Elaboration

Figure 5-9
Elaborations for UOBs 9 and 10

In this example, the analyst has made the elaborations for UOBs 9 and 10 as complete as possible, and will return it to the domain expert for an evaluation. In this interview, the structure of the schematic would be evaluated to confirm that it communicates the expert's knowledge about the scenario. The correctness of the schematics and the elaborations will be confirmed in this process. The review may indicate that some changes need to be made to the captured description. This can take the form of additional objects, activities, facts, and constraints or modifications and deletions to the original lists.

After reviewing the IDEF3 description with the domain expert, the analyst made the following observations which required changes in the IDEF3 Process Schematic.

1. Not all completed purchase requests require authorization signatures.

2. Purchase requests involving direct projects require an authorization signature from accounting and finance to prevent billing material to a

contract that isn't set up to handle material purchases. Indirect projects have no such restrictions.

3. No approvals for purchase requests will be made by account managers without the requester having filled in all the needed information on the purchase request form.

After the review and interview, the new data is evaluated and the lists updated. The additional data is incorporated into the description in the following manner.

The following is a list of the added objects:

- Accounting and Finance Department
- Direct projects
- Indirect projects

The following are the additional *facts* and *constraints*:

- Not all completed purchase requests require authorization signatures.
- Purchase requests involving direct projects require an authorization signature.
- No request will be approved unless a purchase request form has been completed properly.

The additional data and changes suggested by the domain expert are incorporated into the process description (i.e., schematics and elaboration forms). The resulting Process Schematic is illustrated in Figure 5-10.

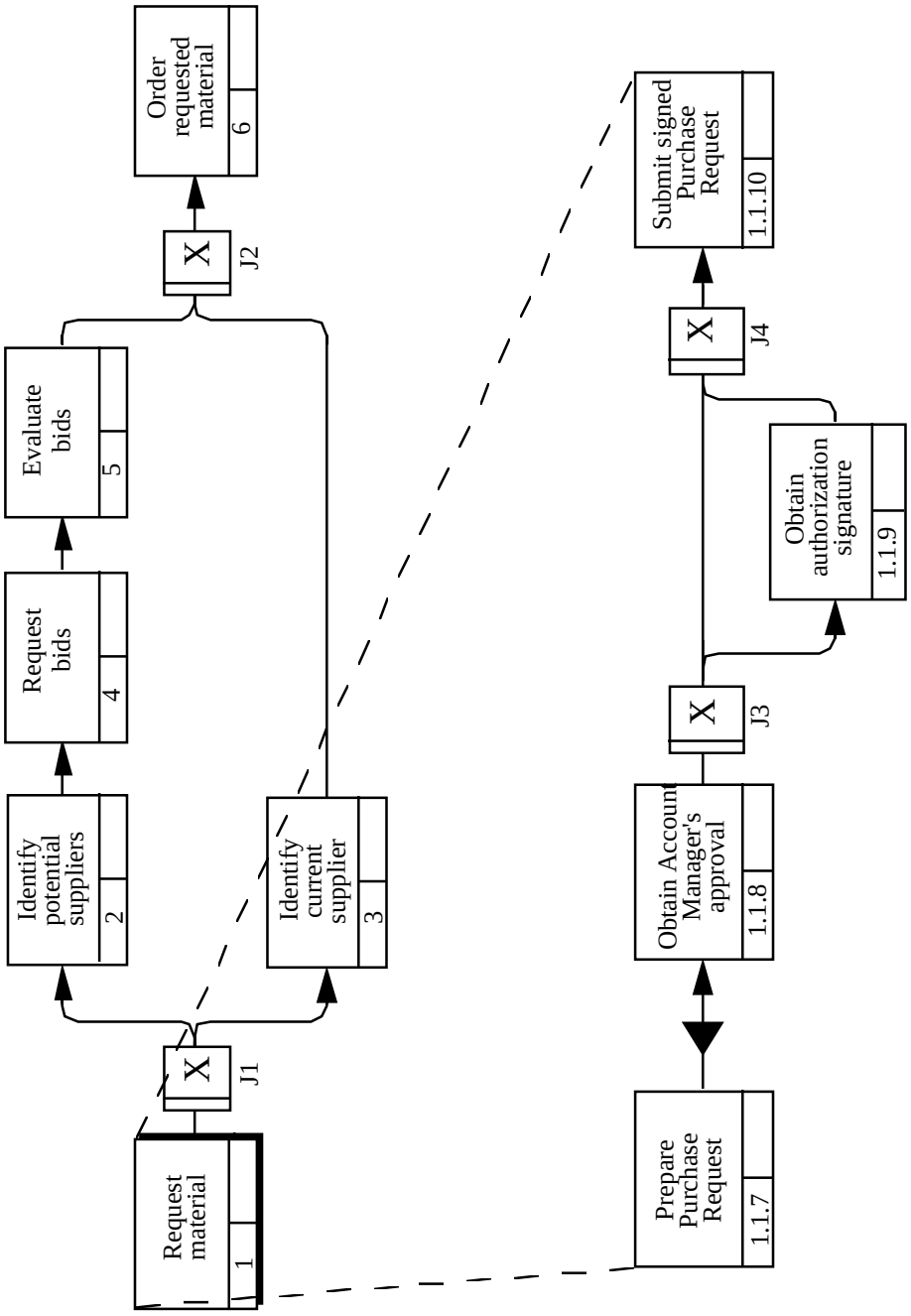

Figure 5-10
Final Process Schematic

The decomposition of the UOB *Request Material* has undergone two structural changes. First, a precedence link with constraints was added to describe the constraint that account managers will not approve a purchase request until the requester has completed the required paperwork. Thus, any instance of the UOB «Obtain Account

Manager's approval» will always be preceded by an instance of the UOB «Prepare Purchase Request.» The second structural change is the introduction of junctions to display the fact that not all purchase requests require an authorization signature.

In the final schematic (See Figure 5-10), the logic associated with junction J3 needs a more detailed explanation (See Figure 5-11). This is accomplished by developing an elaboration for junction J3. On the elaboration form, the label field simply identifies the type of junction. The number field is the number attached to the junction (J3). A junction elaboration form is prepared to clarify the decision logic associated with the junction. In the case of an XOR junction, the junction elaboration allows the analyst to fully describe the rules that determine the choice of a particular path out of the junction.

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Process Description Capture NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING DRAFT RECOMMENDED RELEASED	REVIEWER:	DATE:

Junction No.	
J1	Junction Type: XOR Objects: Facts: Constraints: If there is a current supplier, the company will always purchase the item(s) from that supplier. If there is no current supplier for the needed item, Purchasing identifiers and requests bids from potential suppliers. Description:
J3	Junction Type: XOR Objects: Facts: Some direct contracts aren't set up to handle material purchases. Not all Purchase Requests require an authorization signature. Constraints: Direct projects require an authorization signature. Description: For direct projects, after the Account Manager approves the purchase, the requester must obtain an authorization signature from Accounting and Finance to prevent billing material to a contract that isn't set up to handle material purchases. No other accounts require an authorization signature to proceed with issuing the Purchase Order.

CONTEXT-SETTING REFERENCE: Scenario 1	ITEM DESCRIBED: XOR Junctions J1 and J3	FORM TYPE Junction Elaboration

Figure 5-11
Example Junction Elaborations

Another addition to this process description is an elaboration for one of the links (See Figure 5-10). This link elaboration may not have been entirely necessary in a situation this simple; however, it illustrates how a link elaboration can be associated with a particular link. The link is assigned a number that allows a reader to associate a particular elaboration with the correct link. This link elaboration will contain relevant information to the link between participating UOBs and/or referents.

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Process Description Capture NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

Link No.	Path No.	Source:	Destination:
PL6	PL6.1	Obtain Account Manager's approval (UOB8)	"No op"
	PL6.2	Obtain Account Manager's approval (UOB8)	Obtain authorization signature (UOB9)

Objects: Purchase Request

Facts:

Constraints: (PL6.2) Those authorizing Purchasing Requests must first ensure that the appropriate Account Manager or designated backup has approved and signed the Purchase Request form.

Description:

CONTEXT-SETTING REFERENCE: Decomp 1.1	ITEM DESCRIBED: Precedence Link 6	FORM TYPE Precedence Link Elaboration

Figure 5-12
Precedence Link Elaboration Document

Formulate Object Schematic

To provide a detailed characterization of the objects that participate in a process, it is useful to construct Object Schematics. These are typically developed only for the important objects of the process description. Object Schematics provide a different view of the process being described, i.e., an object-centered view. Object Schematics are most often developed after the Process Schematic; however, some users find it easier to begin with the Object Schematic.

Select Objects of Interest

The first step in formulating an Object Schematic is to select the object or objects of interest. Suppose that in the Purchase Order process the purchase request is the important object. It may be useful to conceptualize the purchase request object as transitioning through several states in the process being described. This would indicate an interest in examining the process from an object-centered view beginning with the initial development of a purchase request through the eventual issuance of a purchase order.

The following example illustrates the process of developing an Object Schematic with this focus.

Identify Object States

Having determined the main object of focus, the analyst begins to develop the Object Schematic by identifying candidate object states. The key object of interest in this case is the Purchase Request (PR), which requires close examination of perceived changes in its state through the process. The terminating state, as indicated by the client, is the point at which the PR finally becomes a Purchase Order (PO). Although there are likely many possible states of a PO, the client's needs dictate no requirement to explore those states.

The first task for the analyst involves identifying candidate object states. A number of sources directly identify these states or provide clues to indicate the possibility that domain experts distinguish certain states. These sources include raw descriptions provided by the domain expert, candidate UOB names, the objects themselves, and UOB elaboration forms. Identifying the possible state changes that an object may undergo in a process often requires that the analyst work with domain experts to either extract or bestow candidate names for object states.

By reviewing the data provided in the interviews, the analyst produces a list of candidate object states like the following list.

PR: Unprepared

PR: Prepared

PR: Approved

PR: Approved requiring authorization

PR: Authorized

PR: Submitted

PO: Issued

This list is likely to undergo change through the identification of logically identical states, through name refinement, and through the identification of previously unnamed or unrecognized states.

Layout Initial Object Schematic

The process of organizing the identified object states into a Transition Schematic occurs as follows:

> **Step 1.** The first step is to identify the initial, or leftmost, object state in the schematic. In this example, the leftmost object state is labeled *PR: Unprepared*.
>
> **Step 2.** The second step is to identify the next state or states to which the object can transition. In this example, the PR transitions from an unprepared state to a prepared state.

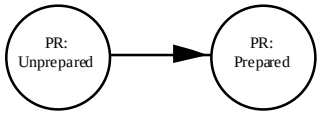

Figure 5-13
Initial Transition Schematic

> **Step 3**. The third step is to repeat steps 1 and 2 until all the possible state transitions are identified.

It is generally helpful to identify and document one transition path at a time before attempting to develop a schematic illustrating all possible paths. In this example, the first point where alternative paths are encountered occurs when a PR is approved by the account manager. At this point, the PR may require authorization. If no authorization is required, the PR may be submitted to Purchasing. This yields the Transition Schematics illustrated in Figures 5-14 and 5-15.

Figure 5-14
Transition Schematic for Path where Authorization is Not Required

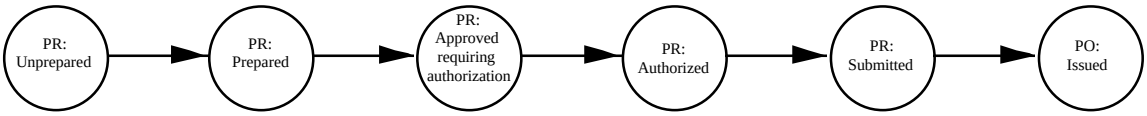

Figure 5-15
Transition Schematic for Path where Authorization is Required

Add Junctions As Required

The fourth step involves combining the two paths into a single schematic by introducing the appropriate logical junction(s).

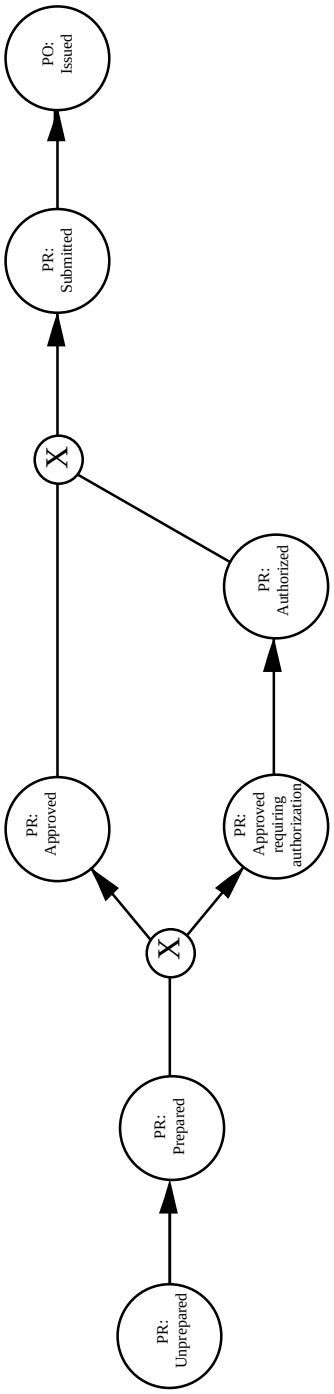

Figure 5-16
Combined Transition Schematic Combining Figures 5-14 and 5-15

Attach Referents

Once the possible paths have been identified and integrated to reflect alternative state transition paths, referents for participating UOBs, scenarios, and other Transition Schematics will be identified and attached to appropriate points on the schematic. This step yields Figure 5-17.

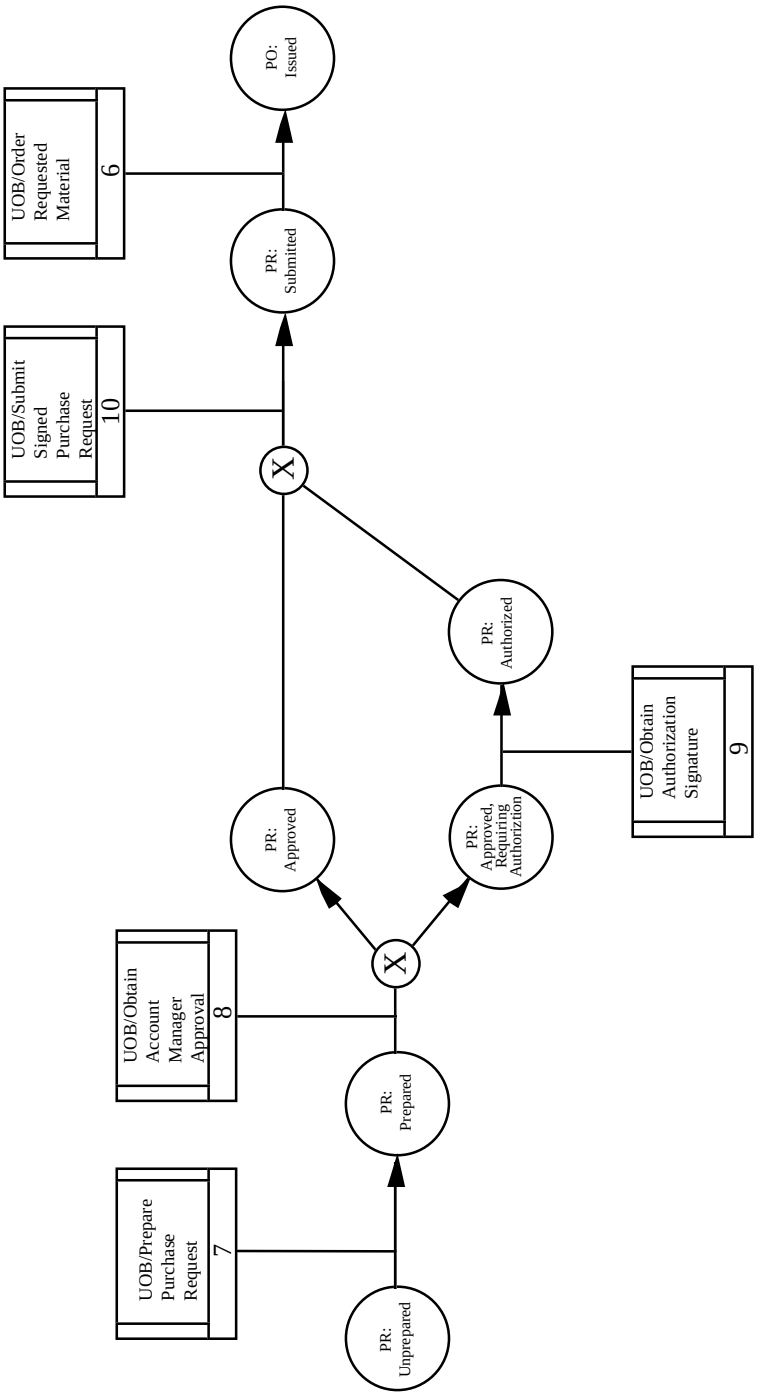

Figure 5-17
Complete Schematic Before First Review

Develop Elaborations

Once the initial Transition Schematic is completed, elaborations must be added to more fully characterize the identified object states as shown in Figures 5-18 through 5-20. In developing the elaborations, the analyst needs to avoid allowing his knowledge of the system type to influence the information placed in the elaborations.

The order in which the elaborations are developed is not necessarily important. It is often useful to develop elaborations in parallel with the rest of the Object Schematic. In particular, concurrent development of the Transition Schematic and associated elaborations may lead to the discovery of previously unidentified states. This example, however, illustrates a situation where the initial elaborations are developed after the Transition Schematic is complete.

USED AT:	ANALYST: I.M. Modeler DATE: 08 Feb 95 PROJECT: Process Description Capture NOTES: 1 2 3 4 5 6 7 8 9 10 REV:	X	WORKING	REVIEWER:	DATE:
			DRAFT		
			RECOMMENDED		
			RELEASED		

Object State No.	
OS2	Object State Name: PR: Prepared Label: PR: Prepared Transitions From Object State(s): PR: Prepared Transitions To Object State(s): PR: Approved PR: Approved requiring authorization Facts: Constraints: State Conditions: The Purchase Request form must include the item description, the number of items needed, the required receipt date (if applicable), the number of the account that will fund the purchase, a written justification for the stated need, and the requester's printed name and signature. Exit Conditions: Other: Description:

CONTEXT-SETTING REFERENCE: Scenario 1	ITEM DESCRIBED: Object State 2 - PR: Prepared	FORM TYPE Object State Elaboration

Figure 5-18
Elaboration for Object State *PR: Prepared*

USED AT:	ANALYST: I. M. Modeler DATE: 08 Feb 1995	X	WORKING	REVIEWER:	DATE:
	PROJECT: Process Description Capture		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10 REV:		RECOMMENDED		
			RELEASED		

Object State No.	
OS4	Object State Name: PR: Approved requiring authorization
	Label: PR: Approved requiring authorization
	Transitions From Object State(s): PR: Approved
	Transitions To Object State(s): PR: Authorized
	Facts: Not all completed Purchase Requests require authorization signatures.
	Constraints:
	State Conditions: Purchase involves use of Direct project funds. Purchase Request form has been signed by the Account Manager or designated backup.
	Exit Conditions:
	Other:
	Description:

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE:
Scenario 1	Object State 4 - PR: Approved requiring authorization	Object State Elaboration

Figure 5-19
Elaboration for Object State *PR: Approved requiring authorization*

USED AT:	ANALYST: I. M. Modeler DATE: 08 Feb 1995	X	WORKING	REVIEWER:	DATE:
	PROJECT: Process Description Capture		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10 REV:		RECOMMENDED		
			RELEASED		

Object State No.	
OS5	Object State Name: PR: Authorized
	Label: PR: Authorized
	Transitions From Object State(s): PR: Authorized requiring signature
	Transitions To Object State(s): PR: Submitted
	Facts:
	Constraints:
	State Conditions: Authorizing official has signed the Purchase Request form. Approving official is not identical with the individual authorizing the Purchase Request.
	Exit Conditions:
	Other:
	Description:

CONTEXT-SETTING REFERENCE:	ITEM DESCRIBED:	FORM TYPE:
Scenario 1	Object State 5 - PR: Authorized	Object State Elaboration

Figure 5-20
Elaboration for Object State *PR: Authorized*

Review Object Schematic with Domain Experts

As with the Process Schematic, the correctness of the Object Schematic and associated elaborations are confirmed through validation with the domain expert. After reviewing the Transition Schematic, the domain expert observes that the allowable state transitions displayed in the schematic do not include those representative of a failed request. Earlier descriptions provided by the domain expert represented the typical case and had not included situations where approval had been withheld or when authorization had been denied. The domain expert's response introduced two entirely new object states.

1. PR: Disapproved

2. PR: Unauthorized

The domain expert also identified transitions through which the identity of the object was preserved and transitions where the object was actually transformed into an entirely

different object. The domain expert's comments to the analyst yield the schematic depicted in Figure 5-21.

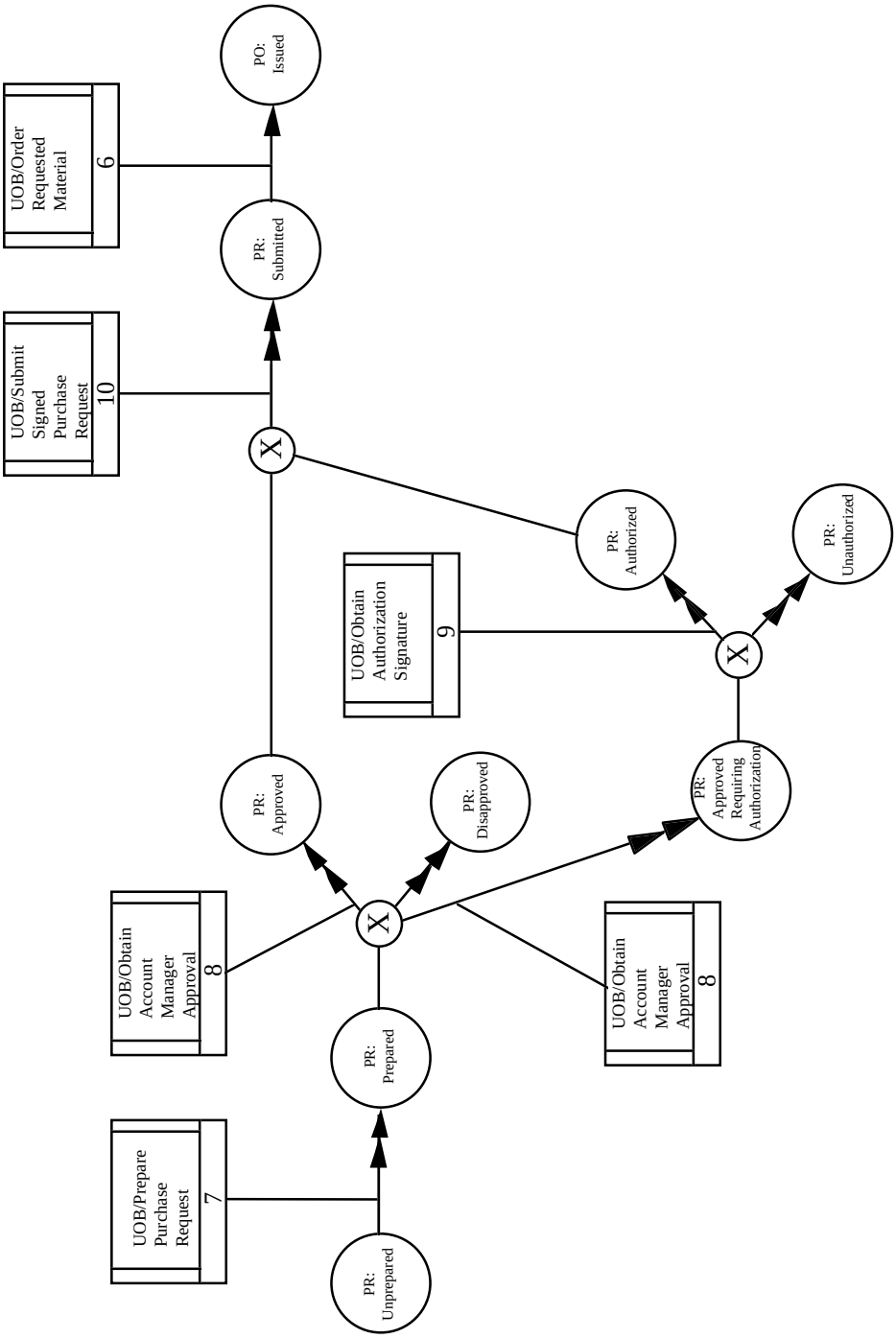

Figure 5-21
Completed Transition Schematic

Additional context-setting information is then added to the Transition Schematic as required. For example, the domain expert's description indicated that purchase requests involving direct projects require an authorization signature. Additionally, the description included discussion of a constraint that account managers or their designated backups must approve all requests involving their projects. This information is noted directly on the schematic. The resulting Object Schematic is displayed in Figure 5-22.

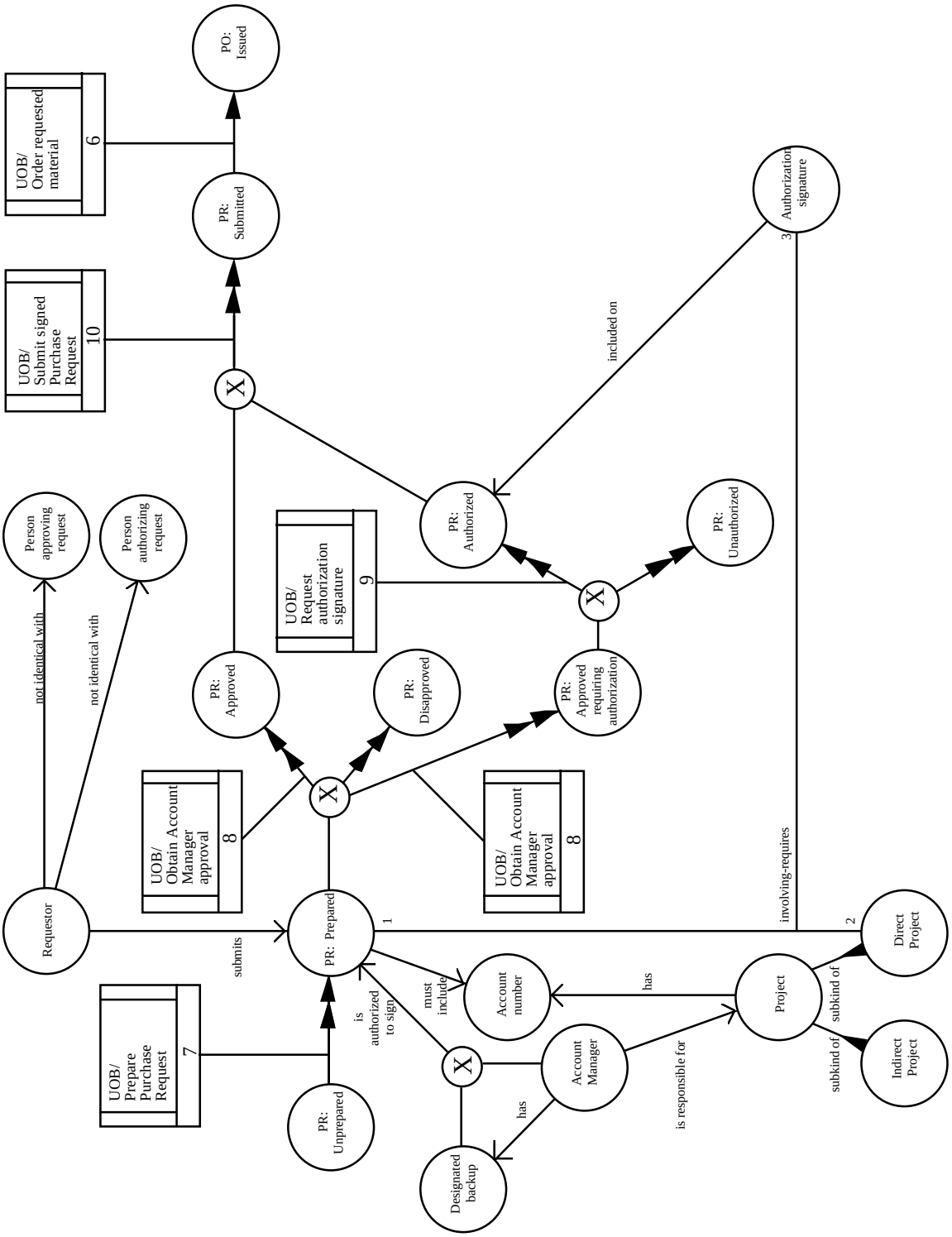

Figure 5-22
Final Object Schematic

UNDERSTANDING IDEF3 PROCESS DESCRIPTIONS

The main purpose of an IDEF3 Process Description is to provide an accurate representation of how a particular system or organization works. An IDEF3 Process Description captures the factual descriptions of the process flow and object state transitions associated with a particular scenario. Reviewers of IDEF3 descriptions may not create them, but must validate the facts in the descriptions. Readers of IDEF3 descriptions may need to acquire knowledge from descriptions that others have created. The general procedure for reading and understanding IDEF3 process descriptions is addressed in this section.

An IDEF3 schematic, whether a Process Schematic or an Object Schematic, is usually read starting with the leftmost element in the schematic. Conventionally, a schematic is read from left to right. To obtain an overview of the described scenario, a mental walkthrough of the schematic is performed. During a mental walkthrough of a Process Schematic, for example, the reader notes precedence relationships and the logical layout of the UOBs. Such a reading will provide a general understanding of the system. Further details of a description may be obtained by reading each UOB and link with their elaborations or descriptions. A comprehensive understanding of the IDEF3 Process Description can be obtained by systematically studying the logic in the schematics.

Description Reading Steps

The facts collected about a system are structured in the IDEF3 Process Description as a set of Process Schematics, Object Schematics, and their associated elaboration language statements. The approach to reading IDEF3 schematics depends on the reader and the amount of information the reader expects to derive.

Because the schematic reading process is highly individualized, it is difficult to express the process in a strict algorithmic format. For example, some people first scan the schematic, then break it up into logical pieces that are easier to understand. In Process Schematics, for example, logical groupings may be created and analyzed to understand the relationships between the UOBs and links in selected portions of the schematic. Once the meaning of the smaller pieces of the schematic are understood, the larger picture becomes evident by taking into account the junctions and their associated logic.

The approach to reading an IDEF3 Process Description can be summarized as follows:

1. Carefully read the statement of purpose, the statement of scope, the objective of the scenario being described, and the viewpoint of the IDEF3 process description.

2. Scan the individual schematic elements (e.g., UOBs, links, junctions, object states) from left to right to gain a general impression of what is being described and to understand generally the structure and logic of the description.

3. Partition the schematic from left to right into logical groupings or structures of schematic elements. Logical groupings are collections of elements that constitute a convenient partitioning of the schematic, enabling systematic review. These groupings most often coincide with process paths (in Process Schematics) or transition paths (in Object Schematics) and may themselves contain logical subgroupings. To achieve a better understanding of the description, these groupings and subgroupings may have to be partitioned in the same manner that the overall diagram was partitioned.

4. Starting with the first element in the left-most grouping, read the schematic from left to right using the following guidelines.

    a. For Process Schematics, read the UOBs and their elaborations. For Object Schematics, read the object states and their elaborations.

    b. Examine the links (precedence links in Process Schematics and transition links in Object Schematics), noting the constraints displayed on the links and the information in the link elaborations.

    c. Study all referents and notes within the bounds of the selected grouping.

    d. Conduct a mental walkthrough of the description, one basic grouping at a time.

    e. When junctions are encountered, follow the paths noting the conditions under which a path will be selected and those under which other paths will be followed.

For more casual readers, a simpler approach is often used. This simpler approach is described in the next section.

Quick Reading of IDEF3 Process Descriptions: An Example

More casual readers of an IDEF3 Process Description will follow a process similar to that described in the preceding section. However, they can expect that as they gain experience in the process, their approach will become personalized. An example approach for reading a schematic is described in the following steps. This outline for reading a schematic would be repeated, with few modifications, for all decompositions, whether found in a Process Schematic or an Object Schematic. In general, decompositions are read after the parent schematic has been read and understood.

The Big Picture

A crucial step in the description-reading process is to understand the big picture relevant to the real-life situation described. This big picture can be gained by reading and understanding the statement of purpose, statement of scope, objective of the scenario being described, and viewpoint of the IDEF3 Process Description. These parts of the description bind the scope of the schematic and tell readers (particularly those familiar with the process being described) what to expect in the top-level schematic. They also indicate the level of detail anticipated.

Scan the Schematic

Readers should become familiar with the scenario by scanning the schematic from left to right. This involves becoming familiar with the individual elements (e.g., UOBs, links, and junctions) displayed in the schematic. This is not an in-depth study of the schematic; rather, it provides readers with a general impression of the process being described and an overall understanding of the logic flow in the scenario.

Understand the Description

In this step, readers gain a detailed understanding of the schematic associated with a scenario, object, or a decomposition of a schematic element. This is the part of the communication process that is most individualized and requires the most time. It is helpful to partition the schematic into understandable pieces. Although there is no «right» or «wrong» way of partitioning a schematic, a partitioning procedure based on the structure of the schematic is often helpful. This approach will be illustrated using an example Process Schematic. Following this example, an illustration of the same concepts applied to an Object Schematic is presented.

The example IDEF3 Process Schematic shown in Figure 6-1 can be partitioned along structural lines as indicated in Figure 6-2.

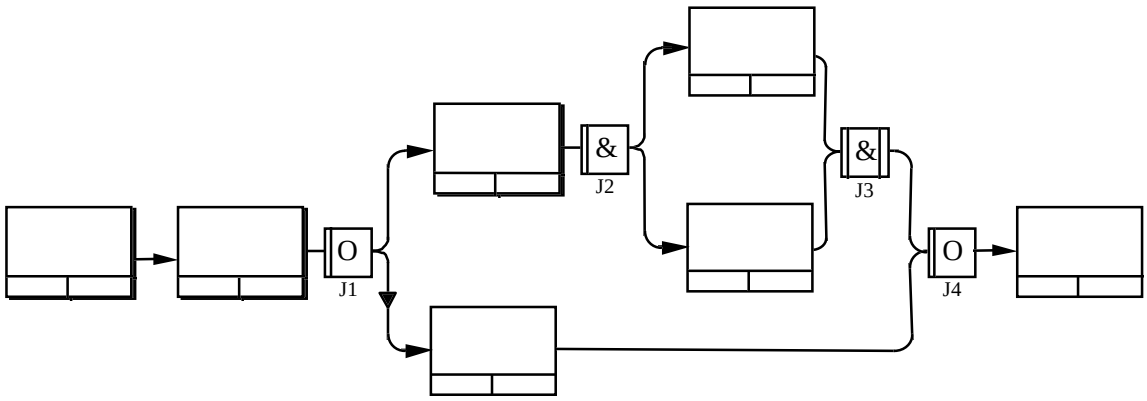

Figure 6-1
Example IDEF3 Process Schematic

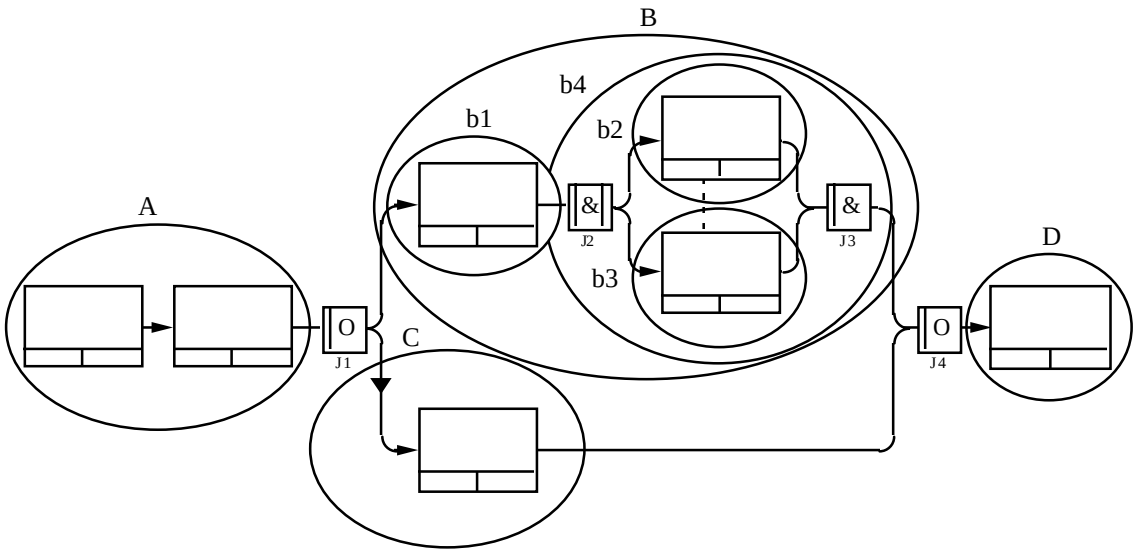

Figure 6-2
Example Partitioning of Figure 6-1

In Figure 6-2, the schematic has been partitioned into four major groupings: A, B, C, and D. These groupings represent four process paths. An examination of these groupings reveals that B can be further partitioned into subgroupings b1, b2, b3, and b4. Once the desired groupings and subgroupings have been established, analysis of the individual structures can begin.

Numbering the groupings 1 through 8 (see Figure 6-3) and starting with grouping 1 on the far left of the description, readers typically proceed from left to right and perform the following activities:

1. Read the UOBs and their elaborations.

2. Examine links and note the information found in the link elaborations.

3. Consider all referents and notes within the bounds of the selected grouping.

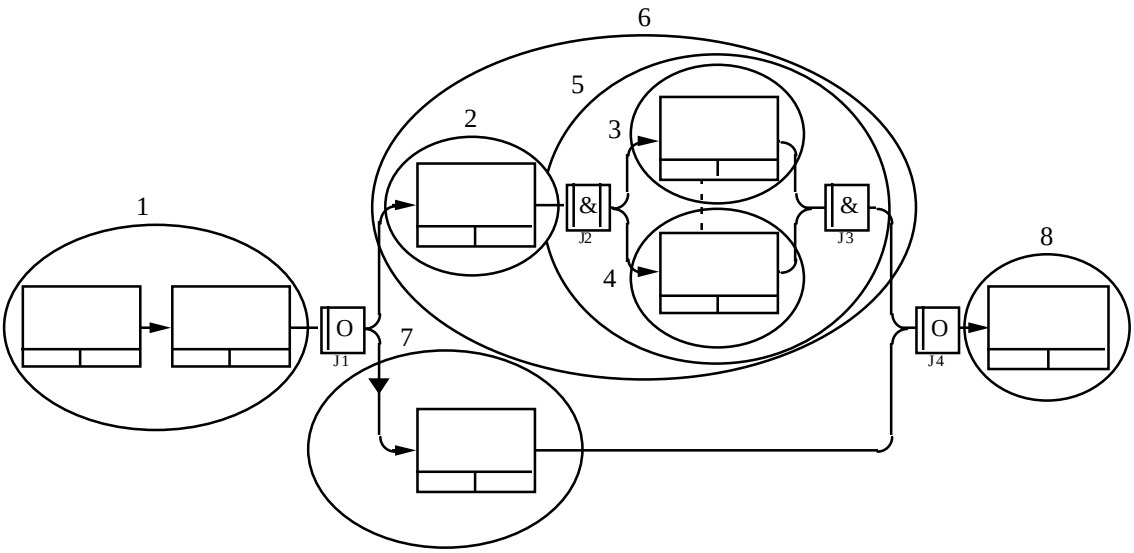

Figure 6-3
Analyzing the Groupings

After understanding grouping 1, the reader will study either grouping 6 or grouping 7. Note that junction J1 is not immediately considered at this stage. Starting with grouping 6, each of the subgroupings 2 and 5 (themselves groupings) will be analyzed. The analysis of grouping 2 means that one UOB and its elaboration must be studied. Grouping 5 is a complex grouping which will be first subdivided into groupings 3 and 4. After completing the study of groupings 3 and 4, grouping 6 is analyzed in its entirety. This process involves understanding the logic of the grouping that includes the J2 fan-out junction from grouping 2 to groupings 3 and 4. To understand junction J2, readers examine the two paths leading from it and note the conditions of flow to these paths. In general, the logic of a junction is analyzed by following all of the paths leading in or out of it, and noting the conditions under which each path will be selected. The study of grouping 5 is completed by analyzing the logic of the fan-in junction J3.

The reader would then attempt to understand grouping 7. After completing the analysis of grouping 7, reading of the description will continue with the analysis of fan-out junction J1. The reader would perform a mental walkthrough of the process starting from grouping 1, noting the conditions under which the flow would branch at the junction

and the conditions governing each fan-out path. In this case, the reader would notice the presence of a constrained precedence link, indicating a need to closely examine the associated precedence link elaboration form.

The next descriptive element of the diagram to be analyzed is the fan-in junction J4 that enables merging of the process paths emerging from groupings 6 and 7. Readers would do a mental walkthrough that involved analyzing the logic of junction J4, noting the conditions under which the two process paths converge.

Finally, grouping 8 is analyzed by reading UOB 8 and its elaboration, and considering any referent that may be attached to it. After this, readers may want to do a complete mental walkthrough of the entire diagram. This will involve starting again at the left end of the schematic and continuing through to grouping 8, considering all the junctions.

The mental walkthrough process for Object Schematics is very similar to that of Process Schematics. The example IDEF3 Object Schematic shown in Figure 6-4 can be partitioned as in Figure 6-5.

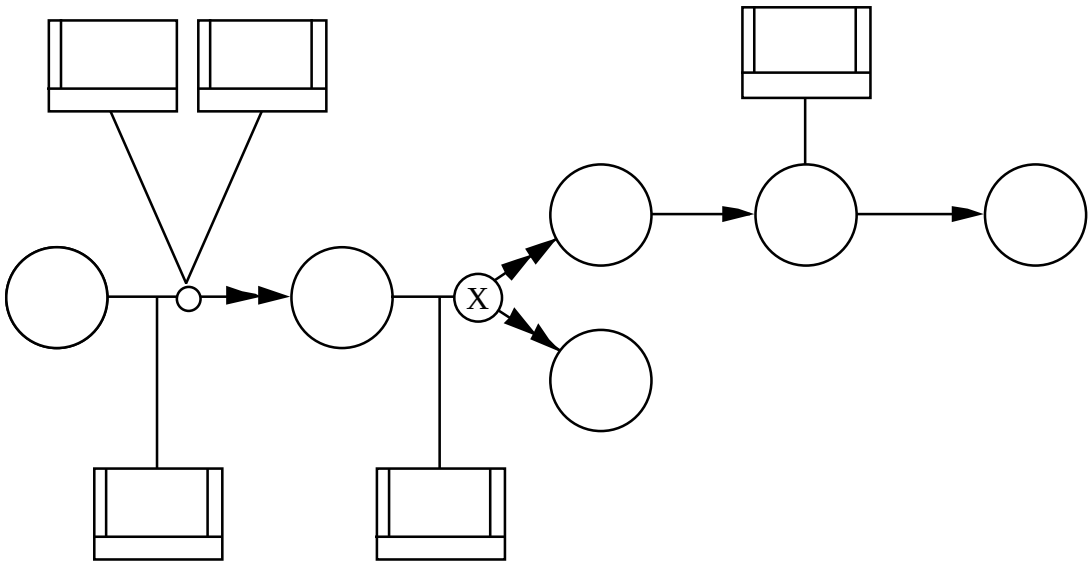

Figure 6-4
Example IDEF3 Object State Transition Schematic

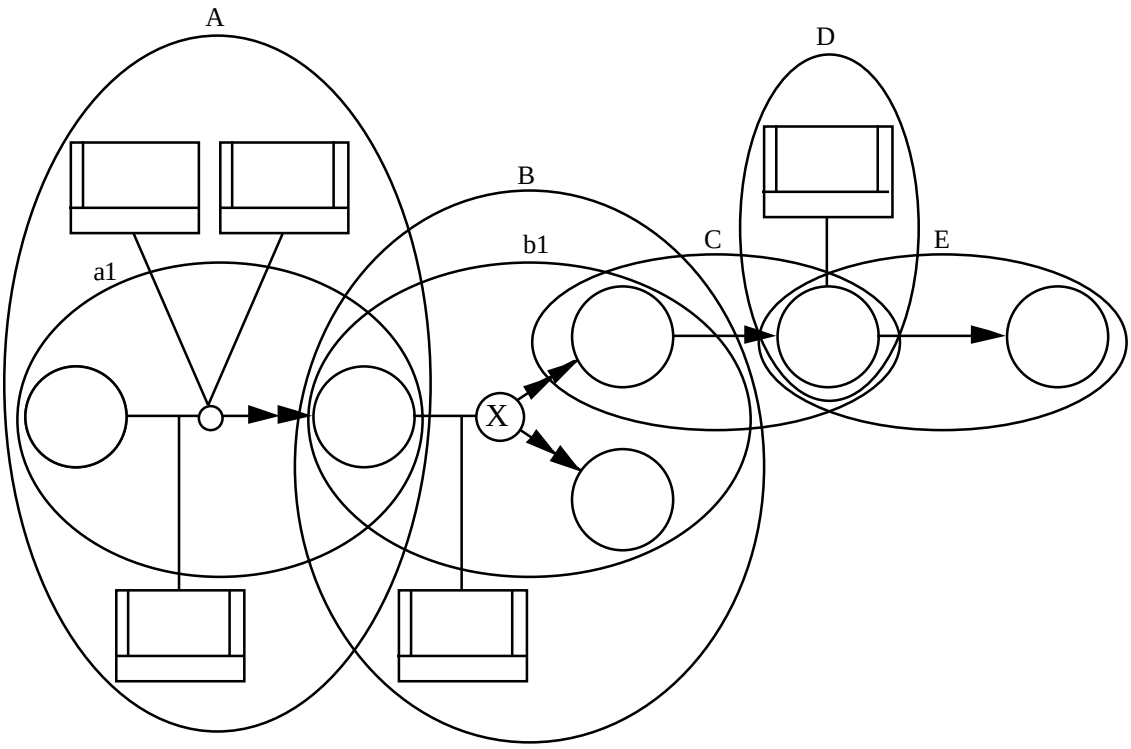

Figure 6-5
Example Partitioning of Figure 6-4

First, the schematic is partitioned into five major groupings: A, B, C, D, and E. For the most part, these groupings center around the four transition links (one of which is a disjunctive junction having two paths). Grouping D centers around a single object state rather than a transition link, treating the state and its attached referent as a single logical unit. Groupings A and B in this example have been further partitioned to include subgroupings a1 and b1.

These groupings can then be numbered as items 1 through 7 (see Figure 6-6). Starting with grouping 1 on the far left of the schematic, readers will typically proceed from left to right while performing the following activities:

1. Read the object states and their elaborations, paying particular attention to state and exit conditions.

2. Examine the transition links and their elaborations, paying particular attention to transition and exit conditions.

3. Consider all context-setting objects, non-transition relations, and notes within the bounds of the selected grouping.

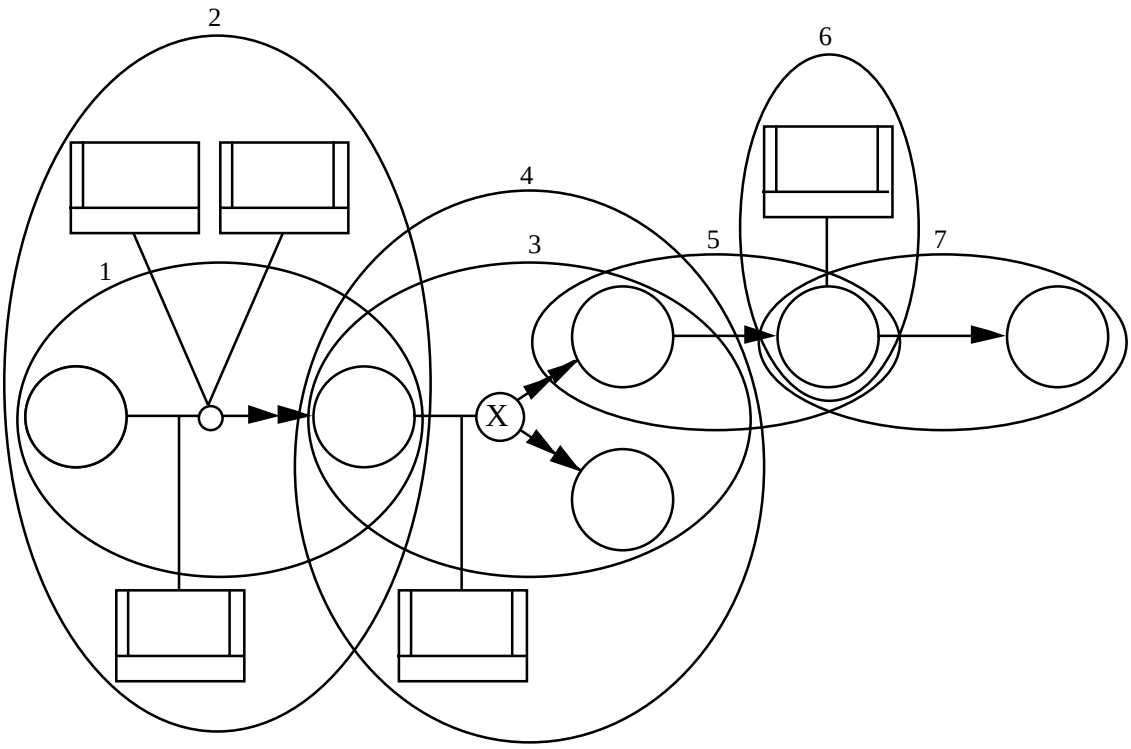

Figure 6-6
Analyzing the Groupings

The process of reading the schematic then proceeds using almost the same pattern as that used for reading the Process Schematic, with a few slight differences. Unlike the partitioning process typically used for the Process Schematic—where groupings and subgroupings tend to form in a strictly hierarchical fashion—Object Schematic groupings tend to overlap with other groupings and to exhibit hierarchical structure. Whereas the context for reading and understanding Process Schematic groupings is established by their place in the hierarchy, the context for understanding an Object Schematic grouping is established by the left-most object state(s) in the grouping. Hence, while proceeding left to right in a schematic, readers will, to some extent, repeat their study of overlapping object states. The first reading of an overlapping object state focuses primarily on the transition and entry conditions associated with the transition link leading to the object state. The second reading focuses primarily on its state and entry conditions. By conducting both readings—where the first reading considers the object state as the right-most element in the grouping and the second reading considers the object state as the left-most element in the neighboring grouping—reviewers gain a clear picture of an object's state transition behavior. As with the Process Schematic, another complete review of the schematic serves to solidify understanding of the description.

BIBLIOGRAPHY

Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23, 123-154.

Allen, J., & Hayes, P. (1987, December). *Moments and points in an interval-based temporal logic*. Technical Report (TR 180), Departments of Computer Science and Philosophy, University of Rochester.

Balci, O., & Nance, R. E., (1987). Simulation model development environments: A research prototype. *Journal of the Operational Research Society*, 38(8), 753-763.

Barwise, J., & Perry, J. (1983). *Situations and attitudes*. Cambridge, MA: MIT Press.

Brachman, R. J. (1985). On the epistemological status of semantic networks. In Brachman, R. J., & Levesque, H. J. (Eds.), *Readings in Knowledge Represenation*. Los Altos, CA: Morgan Kaufmann Publishers, Inc. 191-215.

Coleman, D. S. (1989). *A framework for characterizing the methods and tools of an integrated system engineering methodology (ISEM)* (draft 2, rev. 0). Santa Monica, CA: Pacific Information Management, Inc.

Cross, S. (1983). *Qualitative reasoning in an expert system framework*. Technical Report (T-124). Urbana, IL: University of Illinois Coordinated Science Lab.

Cullinane, T., McCollom, N., Duran, P., & Thornhill, D. (1990). *The human elements of IDEF*. Unpublished manuscript.

D. Appleton Co., Inc. (1985). *Integrated Computer-Aided Manufacturing (ICAM): Information Modeling Manual, IDEF 1—Extended (IDEF1X)* (F33615-80-C-5155), Albany, New York: General Electric Company.

De Kleer, J., & Brown, J. S. (1984). A qualitative physics based on confluences. *Artificial Intelligence*, 24, 7-83.

De Kleer, J. (1979). *Causal and teleological reasoning in circuit recognition* (TR-529). Cambridge, MA: MIT AI Lab.

Devlin, K. (1991). *Logic and information, volume I: situation theory*. Cambridge, MA: Cambridge University Press.

Enderton, H. (1972). *A mathematical introduction to logic*. New York, Academic Press.

Forbus, K., (1984). Qualitative process theory. *Artificial Intelligence*, 24, 85-168.

General Electric. (1985). *Integrated information support system (IISS), volume 5: common data model subsystem; part 4: information modeling manual* (DTIC-A181952). General Electric.

Genesereth, M. R., & Fikes, R. E. (1992). *Knowledge interchange format version 3.0 - reference manual. report logic-92-1*. Logic Group, Stanford University, CA.

Gödel, K. (1931). Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme, I. *Monatshefte Für Mathematik und Physik* **38**, 173-198.

IDEF Users Group. (1990). *IDEF: framework, draft report* (IDEF-U.S.-0001). Dayton, OH: IDEF Users Group.

International Standards Organization. (1987, July). *Information processing systems: concepts and terminology for the conceptual schema and the information base* (ISO/TR 9007). International Standards Organization.

Knowledge Based Systems, Inc. (1991). *Formal foundations for an ontology description method*, (KBSI-SBONT-91-TR-01-1291-02). College Station, TX: Knowledge Based Systems, Inc. (1991).

Knowledge Based Systems, Inc. (1991). *Knowledge based information model integration*. Final Technical Report, NSF SBIR, No. ISI-9060808. College Station, TX: Knowledge Based Systems, Inc.

Knowledge Based Systems, Inc. (1991). *The nature of ontological knowledge: A manufacturing systems perspective* (KBSI-SBONT-91-TR-01-1291-01). College Station, TX: Knowledge Based Systems, Inc.

Knowledge Based Systems, Inc. (1991). *Ontology acquisition method requirements document* (KBSI-SBONT-91-TR-01-1291-03). College Station, TX: Knowledge Based Systems, Inc.

Knowledge Based Systems, Inc. (1991). *Ontology capture tool requirements document* (KBSI-SBONT-91-TR-01-1291-04). College Station, TX: Knowledge Based Systems Inc.

Knowledge Based Systems, Inc. (1991). *Reliable object based architecture for intelligent controllers*. DARPA SBIR 91-050. Contract No. DAAH01-91-C-R235.College Station, TX: Knowledge Based Systems, Inc.

Knowledge Based Systems, Inc. (1992). *IDEF4 method report.* College Station, TX: Knowledge Based Systems, Inc.

Knowledge Based Systems, Inc. (1992). *Ontology capture tool: object-oriented design document* (KBSI-SBONT-91-TR-0292-01). College Station, TX: Knowledge Based Systems, Inc.

Knowledge Based Systems, Inc. (1991). *IDEF5 method report.* Prepared for Armstrong Laboratory, Logistics Research Division, Wright-Patterson AFB, Ohio.

Kripke, S. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica,* 16, 39-48.

Lin, M. J. (1990). *Automatic simulation model design from a situation theory based manufacturing system description.* Unpublished doctoral dissertation, Texas A & M University, College Station, TX.

Link, G. (1983). The logical analysis of plurals and mass terms: A lattice theoretic approach. In R. Bauerle (Ed.), *Meaning, use, and interpretation.* Berlin: De Gruyter.

Mayer, R. J. (1988). *Cognitive skills in modeling and simulation*. Unpublished doctoral dissertation, Texas A&M University, College Station, TX.

Mayer, R. J. (Ed.). (1990). *IDEFØ function modeling: A reconstruction of the original Air Force report*. College Station, TX: Knowledge Based Systems, Inc.

Mayer, R. J. (Ed.). (1990). *IDEF1 information modeling: A reconstruction of the original Air Force report*. College Station, TX: Knowledge Based Systems, Inc.

Mayer, R. J. (Ed.). (1990). *IDEF1X data modeling: A reconstruction of the original Air Force report*. College Station, TX: Knowledge Based Systems, Inc.

Mayer, R. J., et al. (1987). *Knowledge-based integrated information systems development methodologies plan,* (vol. 2) (DTIC-A195851).

Mayer, R. J., Menzel, C. P., & deWitte, P. S. (1991). *IDEF3 technical report*. WPAFB, OH: AL/HRGA.

Mayer, R. J., Menzel, C. P, & Mayer, P. S. (1991). *IDEF3: A methodology for process description,* WPAFB, OH: AL/HRGA.

Mayer, R. J., Edwards, D. A., Decker, L. P., & Ackley, K. A. (1991). *IDEF4 technical report*. WPAFB, OH: Al/HRGA.

Mayer, R. J., deWitte, P. S., Griffith, P., & Menzel, C. (1991). *IDEF6 concept report*. WPAFB, OH: AL/HRGA.

Mayer, R. J., & deWitte, P. S. (1991). *Framework research report*. WPAFB, OH: AL/HRGA.

Mayer, R. J., & Painter, M. K. (1991). *IDEF family of methods*. College Station, TX: Knowledge Based Systems, Inc.

Mayer, R. J., & Wells, M. S. (1991). *Integrated development support environment (IDSE) concepts and standards*. WPAFB, OH: AL/HRGA.

Menzel, C., Mayer, R. J., & Edwards, D. (1994). IDEF3 process descriptions and their semantics. In A. Kuziak, & C. Dagli (Eds.), *Intelligent systems in design and manufacturing*. New York: ASME Press.

Menzel, C. P., & Mayer, R. J. (forthcoming). A situation theoretic approach to the representation of processes. *Proceedings of Working Conference on Models and Methodologies for Enterprise Integration*. New York: Chapman Press.

Menzel, C. P., & Mayer, R. J. (1990). *IDEF3 formalization report*. WPAFB, OH: AL/HRGA.

Neches, R., et al. (1991). Enabling technology for knowledge sharing. *AI Magazine,* 12(3), 36-56.

Overstreet, C. M., & Nance, R. E. (1985). A specification language to assist in analysis of discrete event simulation models. *Communications of the ACM*, 28(2), 190-201.

Painter, M. K. (1990). Modeling with an IDEF perspective: some practical insights. In *Proceedings, SME AutoFact 90*. Dearborn, MI: Society of Manufacturing Engineers.

Pegden, C. D. (1982). *Introduction to SIMAN*. State College, PA: Systems Modeling Corporation.

Poole, D. (1990). A methodology for using a default and abductive reasoning system. *International Journal of Intelligent Systems*, 5, 521-548.

Pritsker, A. A. B., & Pegden, C. D. (1979). *Introduction to simulation and SLAM*. New York: Halsted Press.

Ross, D. T. (1985, June). SADT today: A retrospective on an idea. *IEEE Computer Magazine* (special issue on Requirements Engineering).

(1981). *Integrated Computer-Aided Manufacturing (ICAM): Function Modeling Manual (IDEFØ)* (AFWAL-TR-81-4023, Volume IV). Wright-Patterson Air Force Base, OH: Materials Laboratory, Air Force Wright Aeronautical Laboratories.

(1981). *Integrated Computer-Aided Manufacturing (ICAM): Information Modeling Manual (IDEF1)* (UM 110231200). Wright-Patterson Air Force Base, OH: Materials Laboratory, Air Force Wright Aeronautical Laboratories.

(1981). *Integrated Computer-Aided Manufacturing (ICAM): Dynamics Modeling Manual (IDEF2)* (UM 110231300). Wright-Patterson Air Force Base, OH: Materials Laboratory, Air Force Wright Aeronautical Laboratories.

Soley, R. M. (Ed.). (1990). *Object management architecture guide*. Farmington, MA: Object Management Group, Inc.

Tarski, A. (1956). *Logic, semantics, and metamathematics*. Oxford: Oxford University Press.

van Benthem, J. (1983). *The logic of time*. Dordrecht, Holland: D. Reidel Pub. Co.

Zachman, J. (1986). A framework for information systems architecture. IBM Systems Journal, 26(3), 276-292.

APPENDIX A: IDEF3 ELABORATION LANGUAGE

This appendix contains a detailed description of the IDEF3 elaboration language including a Backus-Naur form (BNF) specification. The IDEF3 elaboration language is broken into two parts: a core language that defines that basic syntactic apparatus and associated logical axioms needed in almost any logical language, and an extension of the core language that includes additional syntactic apparatus, definitions that introduce specialized terms for talking about situations, UOBs and other IDEF3 specific entities, and axioms that constrain their basic use. The core language is loosely based on the Knowledge Interchange Format (KIF) (Genesereth & Fikes, 1992), though a number of constructs—set theoretic ones, in particular—are omitted. In Section A.1, each syntactic category of the core is discussed and examples are given to illustrate how elements of the category are used. A formal BNF is then specified in Section A.2, and a number of definitions and concomitant axioms are provided. An overview of situation theory is provided in Section A.4 to motivate the IDEF3-specific extensions that are introduced. The basic behavior of these extensions is then axiomatized with commentary in Section A.5.

A.1 Description of the Elaboration Language Core

A.1.1 Basic Syntactic Types

The basic syntactic types of the elaboration language core are quite standard. They include the following:

- **Letter:** Letters are the upper and lower case letters of the alphabet.
- **Digits:** The numerals **0** to **9**. Positive digits are the digits from **1** to **9**.
- **Identifier:** An identifier is any string of letters, digits, dashes, and underscores that begins with a letter and ends with a letter or digit.
- **Punctuation:** Punctuation consists of the ASCII characters that are neither letters nor digits.
- **Polarity:** A polarity is either the plus sign «+» or the minus sign «–». These play a special role in terms denoting infons.
- **Posint:** A posint is any sequence of digits of length greater than 0, i.e., intuitively, any numeral that denotes a positive integer. It must begin with a positive digit unless it is simply the digit **0**.
- **Unsigned int:** An unsigned int is the digit **0** or a posint.
- **Int:** An int is either an unsigned digit or a minus sign «–» followed by a posint.

- **Exponents:** An exponent is the letter «**E**» or «**e**» followed by an int.

- **Float:** A float is either an int followed by an exponent, or an optional plus or minus sign followed by an int followed by a decimal point followed by a string of digits followed by an exponent.

- **Number:** A number is either an integer or a floating point.

- **String:** A string is any sequence of ASCII characters (including white space) preceded by and ending with the double quote character «"».

- **Variable:** A variable is any identifier preceded by a question mark «?». The variables of the core IDEF3 elaboration language are completely untyped; no distinction is made even between relation variables and individual variables, a distinction that is drawn in most logical languages. Rather, typing is enforced by means of IDEF3 specific axioms and definitions that introduce the basic semantical categories of the intended semantics language and characterize their properties and the relations between them; see Section A.3.

A.1.2 Operators

Operators are reserved expressions used to form more complex expressions, specifically, terms, sentences, and definitions. The operators that are part of the IDEF3 Elaboration Language are presented in this subsection.

- **Definition Operators:** These operators are used to form definitions for identifiers that are intended to denote an object in one of the basic semantic categories of the language: **define-individual**, **define-function**, and **define-relation**.

- **Term Operators:** These operators are used to form complex terms. There are six such operators: **listof**, **the**, **number-of**, **if**, **cond**, **lambda**, and **kappa**.

- **Sentence Operators:** These operators are used to form complex sentences. The IDEF3 Elaboration Language includes common Boolean and related truth functional operators (**not**, **and**, **or**, **xor** (exclusive disjunction), => (implication), and <=> (co-implication). It also includes the standard universal and existential quantifiers **forall** and **exists**, as well as an array of numerical existential quantifiers **exists-1** (there exists at least one—equivalent simply to **exists**) **exists!-1** (there exists exactly one), **exists-2** (there exist at least 2), **exists!-2** (there exist exactly two), and so on.

A.1.3 Terms

The IDEF3 Elaboration Language supports three types of expressions: *terms*, *sentences*, and *definitions*. A term denotes an object of some sort, though exactly what sort is not determined by the language itself (unlike highly typed languages) but rather separately by means of special axioms (as illustrated below). Terms are divided into the following categories:

- **Atomic Terms:** This category includes all the central simple types of the language, viz., identifiers, variables, ints, floats, and strings.

- **Complex Terms:** A complex term is a term that is built up, ultimately from simpler terms and (perhaps) a sentence and/or operator. The simplest sort of complex term consists merely of a list of identifiers. The grammar of the language allows one to form such terms regardless of what those terms mean. Thus, one could form a term such as, «(Plato Socrates)». However, a complex term of this sort containing n+1 terms is semantically meaningful only if the first term is an n-place function. Thus, in the semantics for the elaboration language «(Plato Socrates)»—assuming the terms «Plato» and «Socrates» mean the philosophers Plato and Socrates, respectively—will be considered meaningless; more precisely, the term will denote a special «empty» individual known as *null*.

Term operators are used to form the other classes of complex terms. **listof** forms a list, or sequence, from its arguments. Thus, the term «(listof 2 3)» denotes the two-element list consisting of the numbers 2 and 3, in that order. Semantically, lists are treated as finite sequences, i.e., functions on proper initial segments of the positive integers.

if and **cond** are used to form *conditional* terms, i.e., terms whose denotations depend on which of several mutually-exclusive conditions hold. Thus, the term

(if (< (age-of ?x) 18) Galen (sister-of Galen))

denotes Galen if ?x is under 18, and it denotes Galen's sister otherwise. Terms formed with operator **cond** work similarly, only they allow the specification of multiple conditions. Suppose that Larry is talking about some unspecified politician. Since there is exactly one such politician that he is talking about (let it be supposed), a corresponding term for that politician can be given a complete definition:

(define-individual POL := (the ?x (and (politician ?x) (talking-about Larry ?x))))

If one knew in addition that the object of Larry's attention had to be either the President, Vice-President, or Speaker of the House (in August 1995), then one could denote the object more specifically by means of the complex term:

(cond (president POL) Bill)

(vice-pres POL) Al)
(speaker POL) Newt),

which picks out Bill Clinton if Larry is talking about the President, Al Gore if Larry is talking about the vice-president, and Newt Gingrich if Larry is talking about the Speaker. Such terms are especially useful in situations where one knows only that the object in a given situation will be one of several objects that can be distinguished by appropriate conditions. Without knowing which in particular it will be, one can nonetheless form a term that picks out the right object depending on which condition holds.

The remaining operators are «variable binding» operators, i.e., they use a variable and some condition involving the variable or variables they bind to specify their denotations. **the** forms a «definite description» term that denotes the unique object satisfying the description in question. So, for example, (the ?x (current-U.S.-president ?x)) denotes (in 1995) Bill Clinton. Note that a necessary and sufficient condition for a definite description term «(the ?x j)»—where j is any «description» (i.e., sentence; see below)—to denote is that one and only one thing that is described by j. Thus, since there are many politicians, the term «(the ?x (politician ?x))» does not denote; or, rather, in the semantics of the elaboration language, it denotes the empty object *null*.

Similarly, **number-of** is used to form terms that denote the number of things satisfying a certain description. Thus,

(number-of ?x (and (integer ?x) (> ?x 0) (< ?x 5)))

denotes the number of integers greater than 0 and less than 5, i.e., the number 4.

lambda and **kappa** are used to form terms that, when semantically meaningful, denote functions and relations. Thus, letting «*» stand for multiplication, the term

(lambda (?x ?y) (+ ?x (* ?y ?y)))

denotes the 2-place function that takes a pair of numbers *m* and *n* to the sum of *m* and the square of *n*. Similarly, the **kappa** operator enables one to form terms for complex properties and relations. Thus, the term

(kappa (?x ?y) (and (person ?x) (house ?y) (owns ?x ?y)))

is the ownership relation that holds between a homeowner and his or her home.

As noted, terms formed from **kappa**, in which **kappa** binds only a single variable, denote properties, or *kinds*. And, because object states are viewed as a subclass of kinds, they too can be thought of as properties. Hence, in these cases, a convention will be to allow the use of **kind-of** to denote kinds and states. Thus, for instance,

(kind-of ?x (and (person ?x) (> (age-of ?x) 40) (< (age-of ?x) 60))

denotes the kind *person over 40 but under 60*. Similarly,

(kind-of ?x (and (person ?x) (sleeping ?x))

is the way the elaboration language expresses the state that would be indicated by *person:sleeping* in a state transition schematic.

An untyped language with operators like **lambda** and **kappa** are, of course, prime candidates for paradox. For instance, the syntax of the language permits the formation of the term

(kappa ?r (forall ?x (<=> (?r ?x) (?x ?x)

which is the infamous Russell property of nonselfexemplification, the property true of exactly those properties that are true of themselves. Such properties can cause trouble, however, only in the context of a logic from which a paradox can arise—in particular, a logic that includes the naive comprehension principle (<=> j[*var*/*term*] ((kappa *var* j) *term*)), where j[*var*/*term*] is the result of replacing all free occurrences of the variable *var* in the sentence j with *term*.[15] We will assume the adoption of some appropriate logic in which such paradoxes are avoided.[16]

A.1.4 Sentences

Intuitively, sentences are expressions that can be true or false. They come in three basic varieties in the elaboration language: *atomic* sentences, *Boolean* sentences, and *quantified* sentences. Atomic sentences are simple sentences consisting of two or more terms. Special cases include equality and inequality sentences, e.g., (= Mark_Twain Samuel_Clemens), and inequality, (/= Shakespeare Roger_Bacon). Similar to functions, any series of terms surrounded by parentheses is a legitimate atomic sentence. However, in order to have any chance of being *true*, such a sentence must consist of an *n*-place relation followed by *n* terms. All other sentences are deemed false in the semantics.

All nonatomic sentences are built up recursively from atomic sentences and logical operators. Complex sentences are of two sorts: standard Boolean sentences, built up using the Boolean and other truth functional operators, and quantified sentences, built up using the quantifiers. Examples of Boolean sentences, playing the role of descriptions in terms, have already been seen above, e.g., (and (person ?x) (> (age-of ?x) 40)). These examples, however, all involve free variables. If these variables are replaced by actual names, they become *closed* sentences, e.g.,

---

[15] An occurrence of a variable *var* in a sentence or term e occurs *free* in e just in case it does not occur within an expression of the form (OP *var* y) or (OP (var_1 ... *var* ... var_n) y) or OP (var_1 ... *var* ... var_n : q) y) in e, where OP is (in the first form) the term-op ‘the’ or one of the numerical quantifiers ‘exists-1’, ‘exists!-1’, ‘exists-2’, exists!-2’, etc., or (in any of the three forms) one of the quantifiers ‘forall’, ‘exists’.

[16] More accurately, avoided as we know. For reasons first uncovered by the famous work of the logician Kurt Gödel, it is not possible to prove (without begging the question) that any reasonably powerful theory is free of contradiction. See [Gödel 31] or, for a more readable account in English, [Enderton 72], Ch. 3.

(and (person Bill) (> (age-of Bill) 40)), which says that Bill—as opposed to some unspecified ?x—is a person over forty.

There are basically two types of quantified sentences: *universally* quantified sentences and *existentially* quantified sentences. These, however, break into several forms. The most general form consists of any quantifier followed by a single variable and a sentence, e.g.,

```
(forall ?x (=>  (and (human ?x) (>= (age-of ?x) 21)
                (adult ?x)))
```

which says all humans 21 years of age or older are adults, and

(exists ?x (and (person ?x) (> (age-of ?x) 100))

which says that there exists a person over 100 years of age.

A second form allows one to bind several variables with a single quantifier. Thus,

```
(forall (?x ?y) (=> (and (B-52 ?x) (F-16 ?x))
                    (weighs-more-than ?x ?y)))
```

says that every B-52 weighs more than every F-16.

The elaboration language provides an optional way to express quantified statements by allowing one to put conditions directly on the bound variables. Thus, the proposition expressed by the above sentence can be expressed a bit more succinctly as

(forall (?x ?y : (and (B-52 ?x) (F-16 ?x))) (weighs-more-than ?x ?y)).

While the unadorned existential quantifier simply states the existence of at least one thing satisfying a certain description, the numerical existential quantifiers state the existence of at least, or exactly, *n* things, for some specific number *n*. Thus, that there are fifty states in the U.S. can be expressed simply as «(exists-50 ?x (state-in ?x US))». Similarly, the fact that there is exactly one U.S. president can be expressed as «(exists!-1 ?x (US-president ?x))».

It should be noted that the numerical quantifiers are, strictly speaking, dispensable. That is, anything that a numerical quantifier expresses can, in principle, be expressed by ordinary quantifiers and the identity relation. Thus, «exists!-1 ?x (US-president ?x))» is equivalent to the sentence

(exists ?x (and (US-president ?x) (forall (?y : (US-president ?y)) (= ?x ?y)))),

i.e., there is at least one U.S. president, and furthermore every U.S. president must be equal to that one. Numerical quantifiers, however, are exceedingly convenient, as to express that some number *n* of things satisfy a certain description requires the use of *n* distinct variables instead of only one.

A.1.5 Definitions

Definitions come in two varieties: *complete* and *partial*. In the case of an individual, a complete definition provides a term that picks out the defined individual. In the case of a function, a complete definition provides a term that picks out the defined function. In the case of a relation, a complete definition provides a set of sentences that jointly specify necessary and sufficient conditions for the relation being defined to hold. A partial definition puts constraints on the individual, function, or relation being defined that (in general) fall short of a full and complete definition. Partial definitions are frequently used simply to introduce a given term into the language with no additional information. The operators**:=** and :=> are used in complete and partial definitions, respectively. In a complete individual definition, the name of the individual and a term defining the individual must be specified. In a partial definition, only the name of the individual must be specified. Optionally, a sentence can be included in a partial definition to restrict the definition of the individual. A constant may have only one complete definition but several partial definitions.

To illustrate, the function *age-of* and the individual *Larry* might be introduced by means of the following definitions.

```
(define-function age-of
    (forall ?x (and (integer (age-of ?x)) (>= 0 (age-of ?x)))))

(define-individual Larry
    (= (age-of Larry) 49).
```

The optional sentence following the (partial) definition of «age-of» specifies that the indicated function must return an integer greater than 0. This function is then used in the partial definition of the term «Larry» to specify his age.

Note that definitions can be used to explicitly introduce terms that denote functions and relations equivalent to those denoted by lambda and kappa terms. Thus, for instance, the property of being a person over 40 but under 60, referred to above by a kappa term, could be named explicitly by an appropriate atomic term by means of the definition

```
(define-relation middle-aged :=
    (and (person ?x) (> (age-of ?x) 40) (< (age-of ?x) 60))).
```

A.2 BNF for the Elaboration Language Core

This section contains the basic grammar for the elaboration language core in extended Backus-Naur form (BNF). The following conventions are used:

- A vertical bar «|» indicates an exclusive disjunction; thus C1 | C2 indicates an occurrence of either C1 or C2, but not both. An absence of such a bar indicates a conjunction.

- A star «*» superscript immediately following a construct (e.g., C^*) indicates that there can be zero, one, or more instances of a the construct.

- A plus sign «+» superscript immediately following a construct (e.g., C^+) indicates that there can be one or more instances of a the construct.

- A construct or combination of constructs surrounded by brackets (e.g., [C1 | C2]) indicates that the construct or combination of constructs is optional.

- In the following grammar, the *terminals* of the grammar—expressions that are reserved in the elaboration language (i.e., that serve a particular purpose in the language)—appear in bold face. Nonterminals—expressions representing categories of expressions—start with "<" and end with ">". For example, the identifier for a variable must start with a question mark. Hence, the construct is shown as: <var> ::= **?**<id>.

A.2.1 Alphabet

The alphabet for the core of the elaboration language consists of the 93 ASCII characters with their standard print representations:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 ( ) [ ] { } < >
= + - * / \ & ^ ~ ' ` ~ " _ @ # $ % : ; , . ! ?

The space character is represented by <space>.

A.2.2 Basic Syntactic Types

<letter> ::= **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**

<posdigit> ::= **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

<digit> ::= **0** | <posdigit>

<id> ::= <letter> [[<letter> | <digit> | **_** | **-**]* <letter> | [<letter> | <digit> | **_** | **-**]* <digit>)]

<punct> ::= **_** | **-** | **~** | **!** | **@** | **#** | **$** | **%** | **^** | **&** | ***** | **(** | **)** | **+** | **=** | **`** | **:** | **;** | **'** | **<** | **>** | **,** | **.** | **?** | **/** | **|** | **[** | **]** | **{** | **}**

<polarity> ::= **+** | **–**

<posint> ::= <posdigit> <digit>+

<unsignedint> ::= **0** | <posint>

<int> ::= <unsignedint> | - <posint>

<exp> ::= **E** <int> | **e** <int>

<float>::= <int><exp> |
<int> . <digit>$^{+}$ [<exp>] |
[**+** | **–**] . <digit> <digit>* [<exp>]

<number> ::= <int> | <float>

<string> ::= **"** [<letter> | <digit> | <punct> | <space> | \" | \\ ]* **"**

<var> ::= **?**<id>

A.2.3 Operators

<operator> ::= <defop> | <termop> | <sentop>

<defop> ;= **define-individual** | **define-function** | **define-relation** | **:=** | **:=>**

<termop> ::= **number-of** | **if** | **cond** | **the**

<boolop> ::= **not** | **and** | **or** | **xor** | **=>** | **<=>**

<quant> ::= **forall** | **exists** | **exists**[**!**]-<posint>

<sentop> ::= <boolop> | <quant>

A.2.4 Terms

<term> ::= <atomterm> | <compterm>

<atomterm> ::= <id> | <var> | <int> | <float> | <string>

<compterm> ::= **(**<term> <term>$^{+}$**) |**
(if <sentence> <term> [<term>]**) |**
(cond (<sentence> <term>**)** **(**<sentence> <term>**)**$^{+}$**) |**
(the <var> <sentence>**) |**
(number-of <var> <sentence>**) |**
(lambda <var> <term>**) | (lambda (**<var>$^{+}$**)** <term>**) |**
(kappa <var> <sentence>**) | (kappa (**<var>$^{+}$**)** <sentence>**)**

A.2.5 Sentences

<sentence> ::= <atomsent> | <boolsent> | <quantsent>

<atomsent> ::= **(**= <term> <term>**) | (/=** <term> <term>**) |** (<term> <term>$^{+}$**)**

<boolsent> ::= **(not** <sentence>**)** | **(and** <sentence> <sentence>$^{+}$**)** | **(or** <sentence> <sentence>$^{+}$**)** | **(xor** <sentence> <sentence>$^{+}$**)** | **(=>** <sentence> <sentence>**)** | **(<=>** <sentence> <sentence>**)**

<quantsent> ::= **(**<quant> <var> <sentence>]**)** |
(forall (<var>$^{+}$ [**:** <sentence>$^{+}$]**)** <sentence>**)** |
(exists (<var>$^{+}$ [**:** <sentence>$^{+}$]**)** <sentence>**)** |
(exists[**!**]-<posint> **(**<var> **:** <sentence>**)** <sentence>**)**

A.2.6 Definitions

<definition> ::= <partial-def> | <complete-def>

<complete-def> ::= **(define-individual** <id> **:=** <term> **)** |
(define-function <id> **(**<var>$^{+}$**) :=** <term> **)** |
(define-relation <id> **(**<var>$^{+}$**) :=** <sentence> **)** |

<partial-def> ::= **(define-individual** <id> **:=** <term> **)** |
(define-function <id> <sentence>***)** |
(define relation <id> **(**<var>***) :=>** <sentence>**)**

A.3 Basic Definitions, Axioms, and Axiom Schemas

The syntax of the elaboration language is highly untyped; sentences and function terms are just lists of terms with no enforced typing. Instead, typing is enforced semantically, in the sense that axioms are provided to ensure that an atomic sentence can be true, or a function term can denote a legitimate object, only if their constituent terms have the right sorts of denotations. This is accomplished by first introducing appropriate semantic categories (i.e., types) and auxiliary notions by means of a series of «define-relation» and «define-function» statements, and then axiomatizing these notions to capture the relevant typing constraints. Note that these initial semantic categories are completely general and independent of IDEF3. Only after these general categories are introduced and axiomatized are IDEF3-specific semantic categories and definitions explicitly introduced and axiomatized (see Section A.5).

It should also be noted that some of the basic semantic categories of KIF, after which the core elaboration language is modeled, are missing, most notably, sets and lists, which are not *explicitly* needed for purposes here. Strictly speaking, set theory is needed to justify definitions, as one must (or at least ought to) be able to prove the existence of the objects one defines (except in the case where one is simply *postulating* or *introducing* an object whose existence cannot be proved from one's given axioms). Their omission here reflects the idea that users should have the option to choose their own set theories; in particular, a user might wish to use one weaker than the powerful von Neumann-Gödel-Bernays set theory of KIF. While acknowledging the need for an associated set theory, commitment to a given set theory is left implicit in this report.

A.3.1 Basic Semantic Categories

In this section, the basic semantic categories *individual*, *function*, and *relation* are introduced, as well as the *null object*, a special object used only to indicate that a term has no genuine denotation.

```
(define-relation individual (?x))

(define-relation list (?x))

(define-relation relation (?x))

(define-individual null)

(define-relation function (?x))
    :=> (relation ?x))

(define-relation interval (?x)
    :=> (individual ?x))

(define-relation integer (?x)
    :=> (individual ?x))
```

Most formalizations involving functions define functions as relations of a certain sort. For instance, in set theoretic treatments such as in KIF, a two-place relation is a set of ordered pairs, and a function is just a two-place relation; such that the first element of any given pair in the relation is not also paired with any other object. Formally, this identification is very convenient. Hence, this identification is added as a defining axiom for functions. Similarly, both intervals and integers are taken to be kinds of individuals.

The duality of functions, viewed as both functions and relations, is captured in the following series of axioms.

Two objects stand in the (binary) functional relation f iff the value of f applied to the first object is the second.

```
(forall (?x ?y ?f : (function ?f)) (<=> (?f ?x ?y) (= (?f ?x) ?y))))
```

Three objects stand in the (ternary) functional relation f iff the value of f on the first two objects is the third.

```
(forall (?x ?y ?z ?f : (function ?f)) (<=> (?f ?x ?y ?z) (= (?f ?x ?y) ?z))))
```

In general, we have the following axiom *schema*:

(forall (?var_1 . . . ?var_{n+1} ?f : (function ?f))

(<=> (?f ?var_1 . . . ?var_{n+1}) (= (?f var_1 . . . ?var_n) ?var_{n+1})))

A.3.1.2 Arity

The arity of a function or relation indicates how many arguments it takes. Because functions are also relations, two separate notions of arity are required: *rel-arity* and *func-arity*. *rel-arity* is a function that takes a relation as an argument and yields a positive integer as its value.

```
(define-function rel-arity
 (forall  (?x ?y : (/= ?y null))
          (=>  (= (arity ?x) ?y)
               (and  (relation ?x) (integer ?y) (> ?y 0)))))
```

Let the lower case Greek letter «n» be the numeral for the number *n*. The following axiom schema expresses the connection between the rel-arity of a relation and the truth of an atomic sentence involving a term referring to that relation, viz., in essence: *An atomic sentence can be true only if it consists of a relation of arity n followed by n terms*.

(forall (?r ?var_1 . . . ?var_n)
(=> (?r ?var_1 . . . ?var_n)
(and (relation ?r) (= (arity ?r) n).

Note that this cannot be expressed by quantifying over numbers with a variable ?n instead using the schematic letter n, as the schematic expression «(?r ?var_1 . . . ?var_n)» is not an expression in the elaboration language proper.

Analogous to *rel-arity*, *func-arity* is a function that takes a function as an argument and yields a positive integer as its value.

```
(define-function func-arity
 (forall  (?x ?y : (/= ?y null))
          (=>  (= (arity ?x) ?y)
               (and  (function ?x) (integer ?y) (> ?y 0)))))
```

The connection between the func-arity of a function the denotation of a function term involving a term referring to that function is captured in the following axiom schema, which says, in essence, that: *An atomic function term that does not denote the null object must consist of an n-place function followed by n terms*.

(forall (?f ?var_1 . . . ?var_n: (=/ (?f ?var_1 . . . ?var_n) null)))
(and (function ?f) (= arity ?f n)))

The following axioms put further constraints on the arity functions.

All relations and functions have a (non-null) arity.

(forall ?x (=> (or (relation ?x) (function ?x))
(/= (arity ?x) null)))

Because every function is a relation, both *rel-arity* and *func-arity* are defined on functions. There is, of course, a close relationship between them, viz.,, if f is viewed as a function, func-arity yields one value on f, and viewed as a relation, rel-arity yields another; more exactly, as the following axiom states,

The rel-arity of a function is one greater than its func-arity.

(forall (?f : (function ?f)) (= (rel-arity ?f) (+ (func-arity ?f) 1)))

An atomic sentence consisting of an n-place relation and some number of argument terms other than n must be false:

(forall (?var_1 . . . ?var_n ?r : (relation ?r) (= (arity ?r) m))
(not (?r ?var_1 . . . ?var_n))), where m is not the numeral for *n*.

A function term consisting of an n-place function term and some number of argument terms other than n must denote the null object:

(forall (?f ?var_1 . . . ?var_n: (function ?f) (= (arity ?f) m))
(= (?f ?var_1 . . . ?var_n) null))), where m is not the numeral for *n*.

A.3.1.2 The Null Object

The following two axiom schemas characterize the null object and its connections with functions and relations.

An atomic sentence that contains a term denoting the null object must be false:

(forall (?r ?var_1 . . . ?var_n : (relation ?r) (or (= ?var_1 null) . . . (= ?var_n null)))
(not (?r ?var_1 . . . ?var_n)))

A function term that includes a term denoting the null object must itself denote the null object:

(forall (?f ?var_1 . . . ?var_n : (function ?f) (or (= ?var_1 null) . . . (= ?var_n null)))
(= (?f ?s_1 . . . ?s_n) null))

A.4 Basic Situation Theory

The underlying intuitive semantics for both process schematics and state transition schematics is based upon situation theory. The purpose of the IDEF3 elaboration language is to permit very precise expression of additional constraints and other information about a given

process or state transition. The core elaboration language is extended with situation theoretic constructs tailored specifically for IDEF3. The theory implicit in these constructs is somewhat less powerful than some versions of full-blown situation theory, but, in addition to a clear, intuitive semantics, experience has shown that this theory is all that is needed for most all enterprise process capture and modeling. In the following sections, a brief overview of situation theory is provided to guide the reader in formulating elaboration statements in the elaboration language.

A.4.1 Situations and Infons

The notion of a *situation* is the most fundamental notion of situation theory. This notion is familiar in the literature of knowledge representation. Situations are (typically) concrete, spatially and temporally extended pieces of the real world, such as a baseball game, a math class, a manufacturing system (though situations in nonconcrete systems are admitted as well; e.g., the field of real numbers). In situation theory, what distinguishes a given situation from any other are the pieces of information it *supports*, or that *hold* in it. In situation theory, individual pieces of information are known as *infons*. The infons in a given domain are themselves constituted by objects, properties, and relations that exist in the domain. (Objects here are construed broadly to include not only physical objects, but also abstract ones like numbers and intervals of time.) More specifically, the *basic* infons in a given situation **s** are the *fundamental units of information*, good and bad, «generated» combinatorially from the relations and appropriate arguments for those relations within s; that is, the basic infons of s consist of all possible legitimate units of information of the form

objects a_1, ..., a_n stand in relation r,

and

objects a_1, ..., a_n do not stand in relation r,

where r and the a_i are all constituents of s. (Relations that hold among individual objects are known as *first-order* relations.) These infons will be represented in the language that will be developed here as «(r a_1 ..., a_n +)» and «(r a_1 ..., a_n -)», respectively. A situation s *supports* a basic «positive» infon (r a_1 ..., a_n +) just in case its component objects a_1, ..., a_n are present in s (at least at some time during s) and stand in the relation r in s, and s supports a basic «negative» infon (r a_1 ... a_n –) just in case a_1, ..., a_n are present in s but do not stand in that relation in s. s *denies* (r a_1 ... a_n +) just in case its component objects a_1, ..., a_n are present in s but do not stand in the relation r in s, and s denies (r a_1 ... a_n –) just in case a_1, ..., a_n are present in s but do stand in the relation r in s. Thus, for example, the infons (mother-of Hillary Chelsea +) and (mother-of Chelsea Hillary –) are supported by, or hold in, typical White House situations s in 1995; by contrast, such situations *deny* (mother-of Chelsea Hillary +) and (mother-of Hillary Chelsea –); we also say that these infons *fail* in such situations. In the language here, these facts would be expressed as «(supports *s* (mother-of Hillary Chelsea +))», «(supports *s* (mother-of Chelsea Hillary –))», «(denies s (mother-of Hillary Chelsea -))» and «(denies s (mother-of Chelsea Hillary +))», respectively, where *s* is an appropriate White House situation.

Note that, because situations are (in general) *limited* pieces of the world, an object b that exists in one situation s may not exist in another s¢. Hence, s¢ will be «silent» on b; more exactly, it will support no information about b. Situations are *partial with respect to information*; they do not answer every question about every individual or every state of affairs. A typical baseball game in the Houston Astrodome, for example, carries no information about, say, the price of Coho salmon in Seattle's Pike Place Market.

To say that s supports a given basic infon (r a_1 ... a_n +) is to say that the individuals a_1, ..., a_n stand in the relation r throughout s. However, things can change within a situation—e.g., one changes from sleeping to waking in typical morning situations. This can be captured in situation theory by counting temporal intervals as individuals, and including a temporal parameter explicitly among the arguments of first-order relations whenever appropriate. Thus, for instance, the property *asleep* will be conceived in fact to be a 2-place relation that holds between individuals and temporal intervals. Thus, if s is a typical morning situation between 6:00 a.m. and 8:00 a.m. at an individual b's house, it is likely the case both that (supports s (asleep b 0600 +)) and that (supports s (asleep b 0800 -)). (If the relevant temporal parameter is understood, then, of course, it can be suppressed as a matter of convenience.) It is presupposed in the semantics of the version of situation theory used with IDEF3 that all subintervals of the interval over which a situation occurs are present in the situation. So, a situation occurring from 6:00 a.m. to 8:00 a.m. supports all relevant temporal information, for example, that the interval from 6:00 to 6:15 precedes the interval from 6:30 to 6:45.

A.4.2 Types, UOBs, and Processes

In most physical systems, one observes multiple occurrences of situations that are *similar* in some respect. In such cases, the similar situations are said to be of the same *type*. For instance, a situation in which Bill Clinton is running on Tuesday and another in which he is running on Wednesday, though perhaps different in many respects, are similar insofar as Clinton is running in them, and are therefore instances of the same type of situation. Situation types are thus general, *repeatable* patterns that can be exhibited by many difference specific situations. This, however, is precisely the character of a UOB in IDEF3, and UOBs are therefore identified with situation types. A situation type is specified in situation theory by an operator that abstracts over similar situations and an appropriate abstraction variable;[17] here we will use the operator «type-of». Thus, the activity just noted is represented as «(type-of ?sit (supports ?sit (running Clinton)))». Similarly, distinct objects can be the same in certain respects, and can be thought of as instances of the same *object type*. Thus, Bill Clinton and Jimmy Carter are alike insofar as they are male politicians; i.e., they are both of the type *male politician*. Thus, *male politician* can be thought of as a property shared by Clinton and Carter, and can be denoted in the elaboration language by «(type-of ?x (and (politician ?x) (male ?x)))». In IDEF3, both situation types and object types can simply be identified with properties—of situations and individuals, respectively.

[17] In situation theory proper, variables correspond semantically to actual «variable objects» in the world, known sometimes as «parameters» or «indeterminates.» For purposes here these entities can be avoided, though there may be certain representational needs that require them.

The operator «type-of» can be understood to be simply a notational variant of the property abstraction operator «kappa» (see Section A.1.3 above).

The importance of types in the context of process capture and process modeling—and, indeed, in the context of modeling generally—is that the semantic content of most all process descriptions concerns *types*. More exactly, a typical process is best thought of as a structured collection of UOBs related to one another in a manner that reflects the *process flow* in a given activation of the process; i.e., the temporal relations between the instances of those types in an activation. For instance, consider the painting process depicted in Figure A-1. This diagram depicts a general process that must begin with an instance of the UOB *Paint Part* (represented by the *Paint Part* box with no predecessor), followed by an instance of *Test Coverage*. At that point, depending on the outcome of the test, an instance of the process can either loop back to another instance of *Paint Part*, or continue on to have the part dried. Thus, there are, in principle, infinite ways this single process can be instantiated by particular courses of events, depending on how many times such a course of events loops back to repeat the *Paint Part* activity.

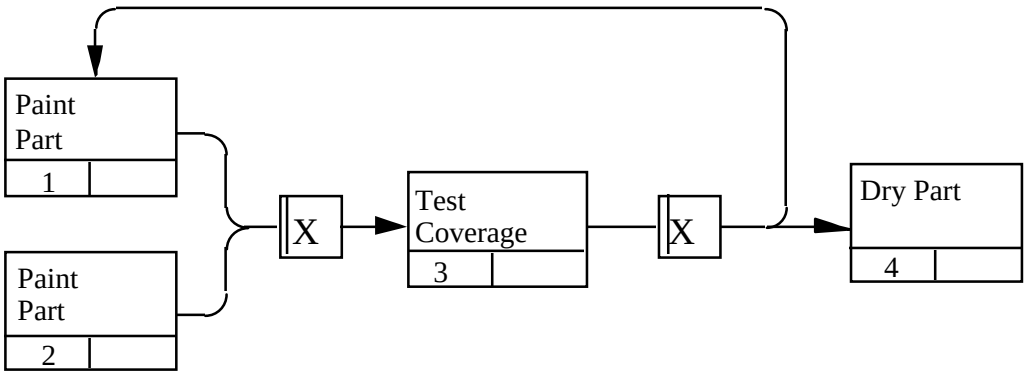

Figure A-1
Paint/Review/Dry Scenario

More generally, a situation type—i.e., a UOB—is specified in terms of a variable «?sit» ranging over arbitrary situations and a formula j specifying the conditions common to situations of that type. Specifically, an activity is referred to by terms of the form «(type-of ?sit j)», read «the type of situation such that j». Thus, recalling the example above, «(type-of ?sit (supports ?sit (running Clinton)))» is read «the type of situation such that it supports Clinton running,» or a little more naturally in this case, «the type of situation in which Clinton is running.» A situation s is of *type* T = (type-of ?sit j) just in case j is true when «?sit» refers to s. If j is of the form «(supports ?sit i)», where i is an infon term, the activity is said to be specified *internally*; otherwise it is specified *externally*. The difference is that an internal specification describes the activity in terms of the infons that its occurrences support, whereas an external specification may refer instead to properties of the activity beyond the infons that its occurrences support, such as, e.g., the causes of its occurrences or the costs involved in maintaining them.

A.4.3 Basic Situation Theoretic Relations

Situation theory is highly *typed* in the sense that the world it describes is partitioned into a number of different *semantic categories*; most notably, objects, first-order properties and relations, infons, situations, courses-of-events, object types, situation types, processes, and temporal intervals. To capture these distinctions, the *theory* of situations developed within the elaboration language defines terms that denote each of these categories. In addition, a variety of terms are defined that signify a class of special relations, along with axioms that express precisely what categories of objects can stand in these relations. With these terms at his or her disposal, a user is able to clearly express any additional information or constraints not expressible in terms of the IDEF3 schematic language.

Specifically, then, the *supports* and *denies* relations between situations and infons were discussed at length above. The *occurrence-of* relation holds between a situation s and a UOB U just in case s is an instance of U. The *activation-of* relation holds between a course-of-events c and the process P just in case c is an activation of P. The *occurs-in* relation holds between a situation s and courses-of-events c just in case s occurs in c. The *activity-in* relation mirrors this relation at the type level—it holds between a UOB U and a process P just in case U is among the situation types that constitute P. The *of-type* relation holds between a situation s and a UOB U just in case s is an instance of U. The *object-in* relation holds between an object b and either a situation s or a UOB U just in case b occurs in s or in instances of U.

A variety of temporal relations are needed to describe the temporal structure of complex processes. The only primitive relation required is *meets*, where, intuitively, one interval *i* meets another *j* just in case the endpoint of *i* is the starting point of *j*. Further relations—e.g., *precedes*, *starts*, *finishes*, *overlaps*, *during*—can be defined in terms of meet, as illustrated in Section A.5.4 below. Note that intervals are treated as first-order objects, thus the temporal relations are all first-order relations. Temporal relations are used to define a variety of corresponding temporal relations among situations.

A.5 A Formal Language for IDEF3 Elaborations

In this section the core elaboration language is extended with definitions that introduce the basic semantic categories of situation theory along with appropriate defining axioms. To aid comprehension, axioms are usually first given in English, and are formatted in italics to enhance readability. Note that this extension of the elaboration language core is *not* a formalization of full blown situation theory. Rather, it is a specification of the basic constructs needed to express situation-based IDEF3 elaborations, as illustrated in the examples above and in Section 3, «IDEF3 Process Description Language,» of this report. Note also that no formal model theory is provided here. Since the purpose of this report is to enable enterprise modelers and knowledge engineers to effectively *use* IDEF3, informal, intuitive characterizations of the semantics of the language have been provided instead.

A.5.1 Extending the Core Elaboration Language

The first task to be addressed is to extend the core elaboration language to the full IDEF3 elaboration language, including a new class of *infon* terms. This is accomplished by adding the following clause:

<infonterm> ::= **(**<term> <term>$^{+}$**)** |
(and <infonterm> <infonterm>$^{+}$**)** |
(or <infonterm> <infonterm>$^{+}$**)** |
(<quant> <var> <infonterm>)

Infon terms of the form **(**<term> <term>$^{+}$**)** are known as *atomic* infon terms.

In addition, the category <compterm> is modified to include the category <infonterm>. For a more robust version, the explicit set theoretic apparatus included in KIF could be added at this point, but this is unnecessary for purposes here; see (Genesereth & Fikes, 1992) for details of the set theory included in KIF.

A.5.2 Basic Situation Theoretic Semantic Categories

In this section the basic semantic categories of situation theory are introduced and characterized.

```
(define-relation infon (?x))

(define-relation situation (?x))

(define-relation UOB (?x))

(define-relation COE (?x))

(define-relation process (?x))

(define-relation interval (?x))

(define-relation polarity)
```

A single axiom expresses that *the basic semantic categories are all disjoint*; e.g., no individual is an infon, function, or relation.

```
(forall  (?x ?y)
   (=>  (and  (or   (= ?x individual) (= ?x infon) (= ?x relation) (= ?x situation)
                    (= ?x UOB) (= ?x COE) (= ?x process) (= ?x polarity))
              (or   (= ?y individual) (= ?y infon) (= ?y relation) (= ?y situation)
                    (= ?y UOB) (= ?y COE) (= ?y process) (= ?y polarity))
              (=/ ?x ?y))
        (forall ?z (not (and (?x ?z) (?y ?z))))))
```

A.5.2.1 First-order Relations

In the version of situation theory developed here, infons are constructed only out of first-order relations and individuals. Hence, the notion of a first-order relation needs to be defined explicitly and axiomatized with an appropriate schema.

(define-relation FO-relation (?rel)
:=> (relation ?rel))

*A **first-order relation** is a relation that can only be true of n-tuples of individuals*.

(forall (?rel : (FO-relation ?rel) (= (arity ?rel) n)))
(forall (var_1 . . . var_n : (?rel var_1 . . . var_n))
(and (individual var_1) . . . (individual var_n))))

Recall that change in the extension of a typical first-order relation over time is captured by including a parameter for temporal intervals among its arguments. For example, the relation *walks* is taken to be a 2-place relation that holds between an individual b and an interval t just in case b walks *throughout* the interval t. Because intervals are themselves individuals, this is accommodated by the above definition.

A.5.2.2 Axioms for Infon Terms

Given the notion of a first-order relation, axioms can be given that express what is required for an infon term to denote a legitimate infon (as opposed to the null object).

A basic infon term denotes an infon if and only if it consists of a term denoting an n-place first-order relation followed by n terms denoting individuals and a term denoting a polarity.

(forall (?rel var_1 . . . var_n ?pol)
(<=> (infon (?r var_1 . . . var_n ?pol))
(and (FO-relation ?r)
(individual var_1)
. . .
(individual var_n)
(polarity ?p))))

A conjunctive (disjunctive) infon term denotes an infon if and only if each conjunt (disjunct) denotes an infon.

(forall (?inf1 ?inf2)
(<=> (and (infon (and ?inf1 ?inf2)))
(and (infon ?inf1) (infon ?inf2))))

(forall (?inf1 ?inf2)
(<=> (and (infon (or ?inf1 ?inf2)))

(and (infon ?inf1) (infon ?inf2))))

Analogous axioms for quantified infons must be stated as a schema. Let *Q* be any quantifier, *var* any variable, and *infterm* be any infon term. If *var* occurs free as the first term in any atomic infon in *infterm*, then

(<=> (infon (*Q var infterm*))
(*Q* (*var* : (FO-relation *var*)) (infon *infterm*))).

Similarly, if *var* does not occur free as the first term in any atomic infon term in *infterm* (i.e., occurs bound, or occurs free elsewhere than as the first term in an atomic infon term in *infterm*, or does not occur at all in *infterm*), then

(<=> (infon (*Q var infterm*))
(*Q* (*var* : (individual *var*)) (infon *infterm*))).

A.5.3 Basic Situation Theoretic Relations

In this section, the basic situation theoretic relations are introduced and, by means of defining axioms, their legitimate argument types are declared. Note that, as with the basic categories of the elaboration language core, «legitimate» here is understood semantically, and enforced axiomatically. Any terms whatsoever can be used to construct syntactically correct *sentences* involving the relation terms introduced below. However, such sentences can be *true* only if the axiomatized constraints are satisfied.

```
(define-relation supports (?x ?y ?z))
    :=> (and (situation ?x) (infon ?y) (interval ?z)))

(define-relation denies (?x ?y)
    :=> (and (situation ?x) (infon ?y) (interval ?z)))

(define-relation occurrence-of (?x ?y)
    :=> (and (situation ?x) (UOB ?y)))

(define-relation activation-of (?x ?y)
    :=> (and (COE ?x) (process ?y)))

(define-relation occurs-in (?x ?y)
    :=> (and (situation ?x) (COE ?y)))

(define-relation activity-in (?x ?y)
    :=> (and (UOB ?x) (process ?y)))

(define-relation of-type (?x ?y)
    :=> (and (situation ?x) (UOB ?y)))

(define-relation object-in (?x ?y)
```

```
    :=> (and (individual ?x) (or (situation ?y) (UOB ?y)))
```

A.5.4 Basic Temporal Relations

In this section, some basic temporal relations between intervals are introduced. The only primitive relation needed for characterizing the temporal intervals used in IDEF3 is the *meets* relation, which can be true only of intervals, as indicated in the following partial definition.

```
(define-relation meets (?x ?y)
    :=> (and (interval ?x) (interval ?y)))
```

Intuitively, as noted, one temporal interval meets another just in case the end point of the first is identical with the starting point of the second. A logic for the *meets* relation as found in (Allen & Hayes, 1987) is assumed [see also (van Bentham, 1983)].

A variety of useful temporal relations can be defined in terms of *meets*. The first is *strongly-precedes*, where one interval strongly precedes another just in case the first meets an interval that meets the second.

```
(define-relation strongly-precedes (?x ?y)
    := (exists (?z ?w : (/= ?z ?w)) (and (meets ?x ?z) (meets ?z ?w) (meets ?w ?y))))
```

Note that, because *points* are intervals, we need to put two distinct intervals between ?x and ?y. In fact, a point can be defined as an interval that meets itself (this is a divergence from Allen and Hayes).

```
(define-relation point (?x)
    := (meets ?x ?x))
```

One temporal interval *i starts* another *j* just in case both are met by a given interval but *j* meets an interval which is met by an interval met by *i*:

```
(define-relation starts (?x ?y)
    := (exists ?z (and  (meets ?z ?x) (meets ?z ?y)
                        (exists ?w (and (meets ?y ?w) (strongly-precedes ?x ?w))))))
```

Similarly, a temporal interval *i finishes* another *j* just in case both meet a given interval but *i* is met by an interval that starts *j*:

```
(define-relation finishes (?x ?y)
    := (exists (?z ?w : (starts ?w ?y)) (and (meets ?x ?z) (meets ?y ?z) (meets ?w ?x))))
```

i overlaps j just in case some interval that finishes *i* starts *j*:

```
(define-relation overlaps (?x ?y)
    := (exists ?z (and (finishes ?z ?x) (starts ?z ?y))))
```

i is *during j* just in case some interval that starts *j* meets *i* and *i* meets some interval that finishes *j*:

```
(define-relation during (?x ?y)
    := (exists (?z ?w) (and (starts ?z ?y) (meets ?z ?x) (finishes ?w ?y) (meets ?x ?w)))
```

Other useful relations can be defined in a similarly straightforward fashion.

A.5.5 The Interval Over Which a Situation Occurs

It is very important in describing processes and their activations to be able to talk about the interval of time over which a given situation occurs. For this reason, a function is defined that, when applied to a given situation, yields exactly that interval:

```
(define-function interval-of
    :=> (forall  (?sit ?t)
                 (=>  (= interval-of ?sit) ?t)
                      (and (situation ?sit) (interval ?t)))))
```

Given the temporal relations, by defining the interval over which a situation occurs, a variety of useful temporal relations among situations can be defined in terms of corresponding temporal relations between the intervals over which they occur. See (Menzel & Mayer, forthcoming) for details.

A.5.6 Using Sorted Variables

Note that the IDEF3 elaboration language proper is completely untyped; in particular, there is only one sort of variable. However, the informal examples shown in Section A.4 and in Section 3 use a wide variety of sorted variables whose possible values are restricted to various semantic categories—e.g., UOBs, situations, infons, etc. This practice can be viewed as simply a convenient use of alternative notation, as any sentence in a so-called many-sorted language with many different sorts of restricted variables can be translated directly into a sentence of a single sorted language such as the elaboration language. The trick is simply to use the terms in the single sorted language that denote the semantic categories to which the sorted variables are restricted in the many-sorted language. For example, suppose «?sit» is a sorted variable ranging only over situations and «?ind» is a sorted variable ranging only over individuals. Then the sentence «(forall (?sit ?ind : (FOO ?sit)) (BAR ?ind ?sit)))» says that, for any situation s and individual b, if s is a FOO, then b bears BAR to s. Clearly, however, this can be expressed in the strict, single-sorted elaboration language as «(forall (?x ?y : (situation ?x) (individual ?y) (FOO ?x)) (BAR ?y ?x))». The use of sorted variables is therefore innocuous, and indeed, encouraged, as their use typically decreases significantly the length of a sentence written in single sorted notation. For a rigorous account of the relation between many-sorted and single-sorted languages, see Chapter 4, §4.3 of (Enderton, 1972). In any case, regardless of whether one is using a many-sorted or single-sorted language, it is generally good practice to choose variables

that reflect their intended semantic categories—e.g., «?ind», «?rel», «?f», «?sit», «?inf», «?uob», «?coe», and so on.[18]

[18] Although this practice is adhered to in informal discussions in this document, it is not always followed in the statement of the formalization above in order to drive home the fact that the elaboration language is itself single-sorted, and that typing distinctions are introduced and enforced axiomatically.

APPENDIX B: IDEF3 GLOSSARY

Activation	A collection of instances of some or all of the UOBs in the process represented by the schematic whose temporal and logical properties satisfy the temporal and logical conditions specified in the schematic. See instance.
Conditions, Entry	Sufficient conditions for an object to enter a state given a (possibly different) object in the source state of the link leading to the destination state that has met the relevant transition conditions. Entry conditions are associated intrinsically with the interface between object states and transition links.
Conditions, Exit	Sufficient conditions for an object no longer to be in the state in question.. Exit conditions are associated intrinsically with object states.
Conditions, State	Conditions that are individually necessary for an object to be in the state in question. State conditions are associated intrinsically with object states.
Conditions, Transition	Conditions that are individually necessary and jointly sufficient for there to be an attempted transition from a source state to the destination state. Transition conditions are associated intrinsically with the interface between object states and transition links.
Constraints	Most generally, a statement which must (or equivalently, must not) hold in a system. Most often, constraints express logical properties of, or connections between, domain objects that must be maintained if the system is to function as intended. Constraints are distinguished conditions known to hold between the objects in a process or between the processes themselves.
Context	The parts of the system under study identified as the bounds within which description development activity will occur and which establish the environment or setting used to document and interpret domain expert knowledge.
Context statement	A written declaration identifying the boundaries of IDEF3 process description development activity (usually expressed in terms of which parts of the system are to be included and which are to be excluded) and the necessary level of detail. The context statement is documented on an IDEF3 Description Summary form.
Decomposition	One of possibly many contextualized descriptions of one UOB in terms of other UOBs. Schematics providing a more detailed view or different perspective of a process with a clearly defined viewpoint.

Description	A recording of facts or beliefs about something within the realm of a domain expert's knowledge or experience.
Domain	A sphere of interest, such as the semiconductor domain or the domain of abstract algebra. A domain has its own distinctive vocabulary for talking about the characteristic kinds of objects and processes typically found in the domain.
Domain expert	An individual considered knowledgeable of, and conversant in, most of the distinguishing characteristics of a certain aspect of a domain. A role played by the primary sources of knowledge from the application domain of interest.
Elaboration	An elaboration provides a detailed characterization of an IDEF3 element (e.g., UOB, Object State, Junction, Link) in a schematic. See Form, Elaboration.
Elaboration language	A structured textual language designed specifically to express process-related information. The IDEF3 elaboration language has the full power of first-order modal logic and set theory.
Facts	Relationships that hold in the actual world. Facts are assertions made about objects.
Form, IDEF3 Description Summary	A structured document that summarizes the evolving/completed process description. It records the project purpose and context and provides a summary of all the schematics and documents used to record the process description.
Form, Elaboration	A structured document used to provide a detailed characterization of IDEF3 elements (e.g., UOBs, object states, links, junctions) in the schematic. Elaboration documents typically include: 1) the element's name, label, and number; 2) lists of the *object types and instances*, *facts*, and *constraints* that are associated with the element; and 3) a textual description of the element.
Form, IDEF3 Schematic	The basic framework for all IDEF3 forms. The IDEF3 Schematic Form is divided into three major sections: 1) Working information (top), 2) Message field (center), and 3) Identification fields (bottom). Working information fields are used to support the kit review process. The identification fields establish the context and purpose of the information on the form. The message field contains the primary message to be conveyed. This field is normally used for schematics, but can be used for any purpose (e.g., glossary, checklists, notes, sketches).
Form, Object Schematic Summary	A structured document summarizing the contents of an IDEF3 Object Schematic.

Form, Pool	A structured document used to list the scenarios, objects, UOBs, and object states identified during description development and provide traceability to the source material supporting those elements.
Form, Process Schematic Summary	A structured document summarizing the contents of an IDEF3 Process Schematic.
Form, Source Material Description	A structured document used in conjunction with the Source Material Log to record more detailed information about each item tracked as source material. In particular, this form is used to capture a concise overview of the main concepts discussed in the source material and provide traceability from the source material to IDEF3 description elements.
Form, Source Material Log	A structured document used to identify and track all data collected during the course of the project. The Source Material Log serves as the primary index to all source material collected and used in an IDEF3 project.
IDEF	Acronym for Integration Definition. Also used to refer to a family of mutually-supportive methods for enterprise integration, including in particular IDEFØ, IDEF1, IDEF1X, IDEF3, IDEF4, and IDEF5.
IDEFØ	Integration Definition (IDEF) method for Function Modeling
IDEF1	Integration Definition (IDEF) method for Information Modeling
IDEF1X	Integration Definition (IDEF) method for Semantic Data Modeling
IDEF2	Integration Definition (IDEF) method for Simulation Modeling
IDEF3	Integration Definition (IDEF) method for Process Description Capture
IDEF4	Integration Definition (IDEF) method for Object-Oriented Design
IDEF4/C++	A specialized Integration Definition (IDEF) method for Object-Oriented Design targeted toward implementation using the C++ object-oriented programming language.
IDEF5	Integration Definition (IDEF) method for Ontology Description Capture
Individual	The most logically basic kind of real world object. Prominent examples include human persons, concrete physical objects, and certain abstract objects such as programs. Unlike objects of higher logical orders such as properties and relations, individuals essentially are not multiply instantiable. Individuals are also known as *first-order* objects.

Instance	As pertaining to an activation, a specific case where one of the pattern of possible activations is exhibited.
Interview	A face-to-face meeting with domain experts to pursue some line of investigation.
Junction	An element of the IDEF3 Schematic Language providing a mechanism to graphically display logical branching.
Kit	An assembly of diagrams, text, glossaries, decision summaries, background information, or any portion of the total IDEF3 description packaged for review and comment. There are three types of IDEF3 kits: object kits, scenario kits, and description kits.
Kit, Description	A compilation from the completed scenario and object kits for a given project containing all the scenarios in the IDEF3 description and their associated documentation. An approved description kit would represent one of the final deliverables in a development effort.
Kit, Object	Kits that address one or more objects and all or part of their associated documentation. The items which may appear in an object kit include Object Schematics, elaborations, etc. packaged for review and comment.
Kit, Scenario	Kits that address one scenario and all or part of its associated documentation. The items which may appear in a scenario kit include Process Schematics, associated UOB decompositions, UOB and link elaborations, etc. packaged for review and comment.
Kit Contents Sheet	An extension to the kit cover sheet used when more space is needed to list the contents of a kit.
Kit Cover Sheet	A structured document that identifies the material assembled as an IDEF3 kit, the review requirements, and an index to the contents of the kit.
Kit Review	A review and approval process used to validate IDEF3 process descriptions.
Link	A syntactic element of the IDEF3 Schematic Language used to connect other IDEF3 syntactic elements. Links denote significant relationships among UOBs, Object States, and Objects. Examples of the types of relations that can be highlighted by IDEF3 links include temporal, logical, causal, natural, and conventional.
Link, Constrained Precedence	A specialization of precedence links that adds further constraints over and above the activation semantics of simple precedence. See Link, Precedence.

Link, Dashed	A syntactic element of the IDEF3 Schematic Language, used in Process Schematics to highlight the existence of a (possibly constraining) relationship between two UOBs. Dashed links carry no predefined semantics. For this reason, they are often referred to as *s* or *User-Defined* links.
Link, First-order	Relations that hold between first-order objects.
Link, Precedence	A syntactic element of the IDEF3 Schematic Language used to express temporal precedence relations between instances of one UOB and those of another.
Link, Relation	A syntactic element of the IDEF3 Schematic Language used in Object Schematics to express additional relations that hold between objects, between objects and object states, between object states, and so forth.
Link, Second-order	Relations that hold between first-order objects and second-order objects, properties, or relations; and relations that hold between second-order objects, properties, and relations.
Link, Strong transition	A specialization of transition links that conveys the additional information that the object involved in the originating state of a transition is the same object as that in the final state of the transition.
Link, Transition	A syntactic element of the IDEF3 Schematic Language used to express the relation *transitions-to* between some source state(s) and some other destination state(s) in an IDEF3 Object Schematic.
Method	An organized, single-purpose discipline or practice for accomplishing some set of tasks. The IDEF methods are specifically designed to accelerate the learning process and help novice practitioners emulate the performance of highly experienced individuals engaged in a particular analysis or design activity. IDEF methods guide users through a disciplined approach, consistent with good-practice experience, to achieve consistently high levels of performance (quality and productivity)
Model	An idealized system of objects, properties, and relations that has been designed to imitate, in certain relevant respects, the character of a given real-world system. Models are idealized systems which are assumed to be «close enough» to provide reliable predictors for the predefined areas of interest within a domain.
Needs statement	A statement that records the source of the request (person or project) and paraphrases the objectives of the project.

Note box	A syntactic element of the IDEF3 Schematic Language that may be used to emphasize the participation of particular objects or relations associated with the IDEF3 element to which it is attached, to tie in specific examples of referenced data or objects (e.g., screen layouts), to highlight special constraint sets associated with a given elaboration, and so forth.
Object	An individual or class of individuals that participate in a process. See *individual*.
Object, First-order	See *individual*.
Object, Second-order	Classes of individuals and first-order relations are second-order objects. See *individual* and *First-order link*.
Object State	An individual or class of individuals that exhibit a specific property or condition (generally indicated by an adjective rather than a common noun). For example, a weapon system development program undergoes a number of different phases that may be viewed as state transitions inaugurated by milestone decisions.
Occurrence	An instance of a UOB within a scenario activation. «Occurrence» is also used to indicate the use of an IDEF3 pool item (i.e., Scenario, UOB, Object, Object State) in some portion of an IDEF3 Process Description. For example, the same UOB pool item may be used more than once in the same Process Schematic.
Process	A real-world event or state of affairs involving one or more individuals over some (possibly instantaneous) interval of time. Typically, a process involves some sort of change in the properties of one or more of the individuals in the process. Sometimes referred to as *process instance*.
Property	An abstract, general feature or characteristic that is multiply instantiable; that is, it can be shared by distinct individuals.
Purpose	The object or end to be attained by engaging in IDEF3 description development activity. That purpose may be simply to document a process—in which case the development of schematics and elaborations is an end unto itself. In most cases, however, IDEF3 description development is undertaken to assist with some discovery or decision-making activities.

Purpose statement	A written declaration specifying the 1) main objective(s) of the effort, 2) needs that the description must satisfy, and 3) questions or findings that the client wants answered. The purpose statement can be separated into two parts, 1) defining a Needs Statement and 2) defining the information goals in terms of how the process description will be used. The purpose statement is documented on an IDEF3 Description Summary form.
Referent	A syntactic element of the IDEF3 Schematic Language used to refer to a UOB scenario or Transition Schematic
Relation	An abstract, general association, or connection that holds between two or more objects. Like properties, relations are multiply instantiable (i.e., it can be shared by distinct objects. The objects among which a relation holds in an instance are known as its *arguments*.
Relation, Temporal	A relation between temporal points or intervals such as *before, during, overlaps* and so forth.
Role, Analyst	IDEF3 expert who is the primary developer of the IDEF3 description.
Role, Commentor	IDEF3 project team member responsible for reviewing draft IDEF3 descriptions and making written critiques.
Role, Librarian	A person assigned the responsibility of maintaining files of documents, making copies, distributing IDEF3 kits, and keeping records.
Role, Project leader	An administrative role that carries the responsibilities for overseeing and guiding an IDEF3 description development effort. In particular, the project leader is ultimately responsible for the outcome of the description development effort, team organization and leadership, and schedule and budget management.
Role, Reader	IDEF3 project team member responsible for reviewing draft IDEF3 descriptions but who is not responsible for providing written comments.
Role, Reviewer	IDEF3 project team member knowledgeable of the application domain and/or the IDEF3 method and responsible for reviewing and/or commenting on draft descriptions and documents. Team members and domain experts can be reviewers. See also *reader* and *commentor.*
Role, Team member	A person involved with the IDEF5 ontology description project.

Schematic	A connected diagram constructed from the lexicon of the IDEF3 schematic language, in accordance with the syntactic guidelines of the language. A visualization, produced using the IDEF3 schematic language, that aids in the construction of process descriptions.
Schematic, Process	An IDEF3 schematic supporting the capture and display of a process-centered view of a scenario.
Schematic, Object	An IDEF3 schematic supporting the capture and display of an object-centered view of one or more scenario(s) of interest.
Schematic, Transition	A type of Object Schematic characterizing the state transitions traversed by participating objects in an instance of the scenario (or process kind).
Schematic, Enhanced Transition	A Transition Schematic that includes context-setting information about the objects and relations that are relevant to the scenario but which do not exhibit the state change behavior of direct interest.
Scenario	In the context of its decomposition, any UOB is a scenario.
Source material	A textbook, a research article, an enterprise-specific document such as a policy manual or a procedure manual, a set of an interview notes, or direct observation notes that has relevant information to the process description development project.
System	A collection of physical and/or conceptual objects that work together to achieve common objective.
Unit of Behavior (UOB)	A term used in IDEF3 to describe types of «happenings.» Concepts such as function, process, scenario, activity, operation, decision, action, event, procedure, and so forth each represent «happenings» involving some circumscribed behavior. The term UOB is used to encapsulate concepts such as these.
UOB Box	A syntactic element of the IDEF3 Schematic Language used to represent a real-world process.
UOB, Child	A UOB in a decomposition.
UOB, Parent	A UOB, acting in the role of a scenario, that establishes the context for a process description.
Validation	The process of checking and ensuring that the IDEF3 process description constructed is both syntactically and semantically correct. A primary means of validating IDEF3 process descriptions is through the review and approval of kits.
Validation, Syntactic	The process of checking and ensuring that the IDEF3 schematic constructed conforms to the grammatical rules of the IDEF3 language.

Validation, Semantic	The process of checking and ensuring that the statements made in the IDEF3 description accurately capture the assertions of the domain expert.
Viewpoint	The perspective taken while examining or describing a system or process. Role-specific and objective viewpoints are captured using IDEF3's UOB decomposition mechanism.