INTEGRATED COMPUTER-AIDED MANUFACTURING (ICAM)
ARCHITECTURE PART III
VOLUME II - PROCEDURES

SofTech, Inc.
460 Totten Pond Road
Waltham, MA 02154

September 1983
Final Report for September 1980 - October 1982

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

DTIC FILE COPY
NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

RICHARD R. PRESTON, Captain, USAF
Project Manager
Computer Integrated Manufacturing Branch
Manufacturing Technology Division

FOR THE COMMANDER

NATHAN G. TUPPER
Chief
Computer Integrated Manufacturing Branch
Manufacturing Technology Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify AFWAL/MLTC, Wright-Patterson AFB, Ohio 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.
The Integrated Computer Aided Manufacturing (ICAM) Architecture Part III was initiated to maintain and update the existing manufacturing architecture as well as develop training courses to assist in the transition of IDEF applications, concepts, and procedures to other Air Force programs. This volume details the Arrow Trace Procedure, IDEF1 Integration Procedure and IDEF1 Integration Procedure.
This report is presented in the following eight volumes:

1. Volume I - Architecture Part III Accomplishments
2. Volume II - Procedures
3. Volume III - Composite Function Model of "Design Product" (DES6)
4. Volume IV - Composite Information Model of "Design Product" (DES1)
5. Volume V - Composite Function Model of "Manufacture Product" (MFG6)
6. Volume VI - Composite Information Model of "Manufacture Product" (MFG1)
7. Volume VII - MFG61 Glossary
8. Volume VIII - Technology Transfer
FOREWORD

This technical report covers the work performed under Air Force Contract #F33615-80-C-5109, "ICAM ARCHITECTURE, PART III," covering the period of September 1980 through October 1982. The contract is sponsored by the Computer Integrated Manufacturing Branch, Manufacturing Technology Division, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, direction of Capt Richard R. Preston. Previous phases were administered under the technical direction of Capt Steven R. LeClair.

Bette Davis is the SofTech Program Manager. The other contributors to this document are as follows:

D. Appleton Company
Rockwell International
SofTech, Inc
SofTech, Inc
SofTech, Inc

Chuck Martin
Richard Heine
Bette Davis
Stan Smith
William St. John
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCOPE 1-1</td>
</tr>
<tr>
<td>1.1</td>
<td>Identification</td>
</tr>
<tr>
<td>1.2</td>
<td>Background</td>
</tr>
<tr>
<td>1.3</td>
<td>Functional Description of Document</td>
</tr>
<tr>
<td>2</td>
<td>REQUIREMENTS</td>
</tr>
<tr>
<td>2.1</td>
<td>IDEF0 Integration</td>
</tr>
<tr>
<td>2.2</td>
<td>Arrow Trace</td>
</tr>
<tr>
<td>2.3</td>
<td>IDEF1 Integration</td>
</tr>
<tr>
<td>3</td>
<td>SHORTENED IDEF0 INTEGRATION PROCEDURE</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.2</td>
<td>Basic Concepts</td>
</tr>
<tr>
<td>3.3</td>
<td>Subsystem Developer Deliverables to the Integration Procedures</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Inputs from the Subsystem Developer</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Identifying Subsystems0 Support of System0</td>
</tr>
<tr>
<td>3.4</td>
<td>Procedure Outputs</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Output Resulting from the Integration Process</td>
</tr>
<tr>
<td>3.5</td>
<td>Integration of &quot;AS IS&quot; Subsystem</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Identification of Subsystem External Arrows</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Summing the Supported Nodes of System0</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.5.3 Maintenance and Summing of the Integration Matrix Form</td>
<td>3-17</td>
</tr>
<tr>
<td>3.5.4 Checking Text and Glossary</td>
<td>3-18</td>
</tr>
<tr>
<td>3.5.5 Comparison of External Arrows</td>
<td>3-21</td>
</tr>
<tr>
<td>3.5.6 Exception Reporting</td>
<td>3-21</td>
</tr>
<tr>
<td>3.5.7 Update of MFGØ</td>
<td>3-21</td>
</tr>
<tr>
<td>3.6 Integrating a Subsystem Model Containing Generic Functions</td>
<td>3-23</td>
</tr>
<tr>
<td>4 ARROW TRACE PROCEDURE</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>4-1</td>
</tr>
<tr>
<td>4.2 The General Arrow Trace Procedure</td>
<td>4-1</td>
</tr>
<tr>
<td>4.3 The Detailed Trace</td>
<td>4-2</td>
</tr>
<tr>
<td>4.3.1 Trace Origin</td>
<td>4-6</td>
</tr>
<tr>
<td>4.3.1.1 Case 1: Label Change</td>
<td>4-6</td>
</tr>
<tr>
<td>4.3.1.2 Case 2: Trace Leads to a Box</td>
<td>4-7</td>
</tr>
<tr>
<td>4.3.1.3 Case 3: Tunneled Arrow</td>
<td>4-8</td>
</tr>
<tr>
<td>4.3.1.4 Case 4: Boundary Arrow</td>
<td>4-8</td>
</tr>
<tr>
<td>4.3.1.5 Error in Trace</td>
<td>4-9</td>
</tr>
<tr>
<td>4.4.1 Trace Target</td>
<td>4-9</td>
</tr>
<tr>
<td>4.4.1.1 Case 1: Label Change</td>
<td>4-10</td>
</tr>
<tr>
<td>4.4.1.2 Case 2: Trace Leads to Box</td>
<td>4-11</td>
</tr>
<tr>
<td>4.4.1.3 Tunneled Arrow</td>
<td>4-11</td>
</tr>
<tr>
<td>4.4.1.4 Boundary Arrow</td>
<td>4-11</td>
</tr>
</tbody>
</table>
### TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 Examples from MFG0 Arrow Trace</td>
<td>4-11</td>
</tr>
<tr>
<td>4.5.1 Label Changes</td>
<td>4-12</td>
</tr>
<tr>
<td>4.5.2 Trace Leads to a Box</td>
<td>4-12</td>
</tr>
<tr>
<td>4.5.3 Tunneled Arrow</td>
<td>4-12</td>
</tr>
<tr>
<td>4.5.4 Boundary Arrow</td>
<td>4-12</td>
</tr>
<tr>
<td>5 IDEF1 INTEGRATION PROCEDURE</td>
<td>5-1</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>5-1</td>
</tr>
<tr>
<td>5.2 Phase Zero</td>
<td>5-3</td>
</tr>
<tr>
<td>5.3 Phase One</td>
<td>5-6</td>
</tr>
<tr>
<td>5.4 Phase Two</td>
<td>5-11</td>
</tr>
<tr>
<td>5.5 Phase Three</td>
<td>5-16</td>
</tr>
<tr>
<td>5.6 Phase Four</td>
<td>5-24</td>
</tr>
<tr>
<td>5.7 Conclusion</td>
<td>5-26</td>
</tr>
</tbody>
</table>
SECTION 1

SCOPE

1.1 Identification

This volume documents the new IDEF procedures developed since June of 1981. Later volumes of this report present models of the functions or the information used in aerospace design and manufacture. Those models have been modified, extended or improved using the procedures documented herein.

This volume describes the procedures used in evolving the Architecture of Manufacturing and design as they currently exist. It does not present the models which make up the architecture.

Sections 3 and 4 of this volume replaces Appendix B of AFWAL-TR-81-4023, Volume III, "Integration using Architecture" published in June of 1981 as part of the ICAM Architecture Part II, it is additionally, an expansion to AFWAL-TR-81-4023, Volume IV "Function Modeling Manual."

Section 5 of this volume is an adjunct to Volume V Information Modeling Manual of the same 1981 report.

This Volume documents work performed under ICAM Project Priority 1104 - ICAM Architecture of Manufacturing Part III.

1.2 Background

The use of the IDEF methodologies on ICAM projects, Air Force Technology Modernization (Tech Mod) programs and similar DoD modernization programs has resulted in an overall need for cost effective and standardized procedures dealing with model integration and validation. This need was first formally addressed during the ICAM Architecture Part II Project in which the Function Model of "Manufacture Product" MFG was integrated with two subsystem models. The procedure used and the results obtained are documented in AFWAL-TR-81-4023, Volume III "Integration Using Architecture."

Through these early integration efforts and through experience gained using the architecture models in conjunction with Tech Mod Programs, recommendations for improvements and additions to these procedures were made by developers and users.

Therefore the ICAM Architecture Part III Project established three procedures aimed at reducing both the costs and time required for integration of subsystem models and validation of resulting composite models. These new procedures are documented in this volume.
1.3 Functional Description of Document

This volume (II) documents procedures used in the development of the architecture of design and manufacture. That architecture appears in other volumes of this report: Volumes III-DES0, IV-DES1, V-MFG0, VI-MFG1.

This volume is intended as a guide for the development of IDEF0 models by manufacturing analysts and industrial engineers involved in the integration of new manufacturing and computer system technology into the production environment. It provides a common baseline for communication and decision making during the "Understanding the Problem" phase of such projects. It can be used by management and engineer's to identify the areas impacted by proposed changes and introduction of new technologies.

Experience, from current Technology Modernization Programs, has shown that the function model MFG0 can serve as either a guide for model development or be annotated to provide a company specific architecture.
2.1 IDEF0 Integration

During the period in which the architecture of design and manufacturing was developed, several subsystem architectures were developed. These included MCMM (Manufacturing Control and Material Management), and SMC (Sheet Metal Center), and QA (Quality Assurance).

The SMC and MCMM subsystem IDEF0 models were originally related to the composite IDEF0 model of manufacturing (MFG0) using the procedure documented by Appendix B of AFWAL-TR-81-4023, Volume III "Integration Using Architecture" published in June of 1981. Their support of MFG0 was then documented using the procedure given in Section 3 of that volume. The integration QA0 into MFG0 and DES0 followed the procedure of Section 3 of this report in its entirety.

The original procedure is completely valid. The new procedure was developed to provide the documentation of support arrows on the IDEF0 composite view and to provide a less cumbersome method while retaining most of the benefits of the original procedure. The main purpose of the new procedure was to reduce the time and manhours expended on integration efforts.

2.2 Arrow Trace

As part of the integration of subsystems into the Manufacturing Architecture (Subsystem0 into MFG0), a more complete form of arrow definition known as an "arrow trace" was developed and applied to MFG0. This new procedure incorporates the formerly developed glossary definitions of arrow labels, adding the following additional information to the textual definitions to validate and verify consistency in arrow data:

- A list of synonymous terms used for the data carried on the arrow.
- A list of source functions which generate the data carried on the arrow.
- A list of target functions which utilize the data carried on the arrow.
- A list of the sub-parts (origin components) comprising the data carried in the arrow, as shown by the arrow branching and joining structure.
- The name of the more inclusive data item or items (usage components) which contain the data carried on the arrow, as shown by the arrow branching and joining structure.
This procedure has been needed in the application of the integration procedure in order to document the complete impact of a subsystem on the total Manufacturing Architecture. Also, the arrow tracing procedure has been found to be helpful in pointing out modeling errors and inconsistencies in the arrow structures, such as inconsistent use of arrow labels.

2.3 **IDEF1 Integration**

The IDEF1 integration procedures were developed to meet a need equivalent to that met by the IDEF0 integration procedures.

The procedures were used to extend the IDEF1, model of manufacture (MFG1).

The subsystems integrated were Integrated Center (ICENT), Integrated Planning System (IPS) and Quality Assurance (QA).
SECTION 3
SHORTENED IDEF0 INTEGRATION PROCEDURE

3.1 Introduction

The shortened IDEF0 integration procedure discussed in this document is a specific phase in an integration process which is intended as an on-going aid to the developers and potential users of newly developed subsystems. The complete process is portrayed in Figure 3-1.

The complete process consists of three phases:

1. Scoping
2. Integration of the "AS IS" subsystem model
3. Integration of the "TO BE" subsystem model

Phase one, which precedes the phase discussed in this procedure, provides for a general scoping of the subsystem developers task. Before development of a new subsystem is initiated, the nodes in the existing System0 to be replaced or supported by the subsystem are identified. This list of nodes provides the contracting office and the developer with a clear specification of the scope of development to be undertaken.

The list of nodes defines the area to be further documented by the developer's "As Is" model.

The definition of any node may be further refined by:

- further detailing or decomposing of the node
- identifying that specific arrows are added, deleted or changed in the context of the node.

Phase two, which this procedure discusses -- when the subsystem developer has completed an "AS IS" model -- specifies a comparison of functions and external interfaces between the subsystem model and the existing "AS IS" System0. The comparison is not exhaustive, and discrepancies noted need not be corrected immediately. The list of discrepancies is used as a guide by the subsystem developer in developing his "TO BE" specifications and by the integration team for review at the next level of integration effort.
Figure 3-1. Shortened IDEF0 Integration Procedure
In the final phase, after this procedure is completed and when the subsystem developer has completed a "TO BE" IDEF0 specification of his subsystem, the comparison of functions and interfaces is repeated with greater rigor and is extended to an identification and consideration of functions which are related to, but not included in, the subsystem. Such functions are considered so as to obtain greater precision and rigor in the specification of Subsystem0 to System0 interfaces. Analysis of the interfaces may indicate a need to change areas of the architecture outside the subsystem to accommodate revised needs or outputs resulting from subsystem installation.

This final phase uses both as-is and to-be versions of System0 since new subsystems must meet two integration criteria. That is, the new subsystem must be useful in factories as they exist today and must also fit smoothly into an image ("TO BE" model) of the updated and integrated factory of tomorrow.

It is within this total integration scenario that this procedure is designed to operate.

Figure 3-1 shows an overview of the total process just described. This illustrates the ultimate purpose and intended outputs of the process of which this procedure is a part. The portions of the process covered by this procedure deal with "AS IS" models and are the primary part of Phase II.

3.2 Basic Concepts

In its simplest form, integration using IDEF0 would involve the replacement of a function represented by a single IDEF0 box by another IDEF0 box. For such replacement to be accepted,

- The new system must be able to use the same information now being supplied to the function.

- The new system must be able to supply the same information now being supplied by the function.

- There must be agreement that the processing performed within the new system is at least equivalent to the processing within the existing system.

The first requirement could be checked by reviewing the input and control arrows of the new and of the old IDEF0 box. The second requirement could be checked by reviewing output arrows. The third requirement could be checked by a discussion of the two box labels. Review of the box labels could be supported by examining any diagrams which detailed the two simple boxes. The procedure in the remainder of this report deals with the requirements just presented, but applies them to the more complex situation which normally exists.
In practice, the parts of an existing system to be replaced by a new subsystem rarely appear as a single box in the architecture of the existing system. Also, differences in implementation methods, terminology or in grouping of data into pipelines may cause difficulty when IDEF0 arrows are compared between models. This latter difficulty could occur even when only one box from each system is being examined.

3.3 **Subsystem Developer Deliverables to the Integration Procedure**

3.3.1 **Inputs from the Subsystem Developer**

The subsystem developer is responsible for providing:

**Initial Input**

a) Subsystem Statement of Work to be performed.

b) A textual description of the project including:

   - An expanded discussion of the purpose and viewpoint of the model
   - A summary of the types of improvements which will be sought during the development of the subsystem (see Figure 3-3).

c) A matrix showing, for each lowest level box in the "AS IS" subsystem IDEF0 model (Subsystem0) the node or nodes of System0 which it supports. The form used is illustrated in Figure 3-2. Preparation of the form by the subsystem developer is discussed more fully in paragraph 3.3.2.

d) Copies of the final "AS IS" IDEF0 models created during the development of the subsystem. The IDEF0 model must include:

   - A node diagram
   - The complete hierarchy of diagrams
   - Related FEO's
   - Texts for all diagrams
   - Glossary covering box names and arrow labels whose meaning is not self-evident.
3-5

e) Responses to all exceptions raised by the integration team during the ongoing integration effort.

3.3.2 Identifying SubsystemΩ Support of SystemΩ

This step is carried out by the subsystem developer. For each box in SubsystemΩ which is not decomposed,

1) Analyze the SubsystemΩ diagram, text, and glossary relating to each "lowest-level" Node Number.

2) Review the Node Diagram and individual diagrams for the SystemΩ, to locate a node which performs a function similar to the SubsystemΩ function.

3) Search for any additional matching nodes in the SystemΩ until all nodes have been reviewed.

4) Read and study the SystemΩ diagrams in light of the matches made in Steps 2 and 3, including the "parent" diagrams of SystemΩ matching nodes as well as any glossary and text.
PURPOSE: This model will be reviewed with system analysts to define ways in which computers could assist in some of the duties of a typical foreman in an aerospace manufacturing company. It therefore stresses the view of a typical foreman of his current activities with all contradictory and irritating factors shown. The presence of the foreman to perform the functions is assumed, but concerns for personnel hiring, training, etc., and for obtaining equipment and facility maintenance are not included. The model will be annotated with mechanism arrows to show which functions will appear in the next--functional--spec of a computer system.

VIEWPOINT: The view of the professional foreman is assumed. Any machine is assumed to be generic as are employees, moves, budgets, etc. The existence and functioning of the department in a physical sense is shown at or above the A-1 level only. A-0 and lower diagrams deal with messages from and messages to the foreman’s environment.

CONTEXT: This project will develop computer software to be run on a minicomputer dedicated to each foreman. The computer will be used to track cell load at the operation level, the assignment and expected availability of each operator, set-up man, and machine and the status of material handling equipment. Based on this knowledge, the program will compute the result of various options considered by the foreman and will store the results of his decision. The system will operate in real time and will give notice of upcoming or missed milestones. The program will track cell inventory. Links will be available for later networking of the minicomputers to provide for coordination from a center control program.

Figure 3-3. Summary of Improvements

5) Record each match identified in Steps 2 and 3 by entering a dot (.) on the intersection of the appropriate row and column of the Integration Matrix Form.

6) Identify any adjustment in context needed to show what parts of the box are supported. Assume for example, that the decomposition of the supported box appearing as item (a) in Figure 3-4 would look like item (b) of Figure 3-4. Then the environment must be redefined to agree with item (c) of Figure 3-4 which now only displays the supported output labelled "p." The output "n" has been dropped because it is not within the context supported by the subsystem mechanism arrows.
The maintenance and revision of this form after it is submitted is the responsibility of the integration team. The procedure for this maintenance and revision is discussed in paragraph 3.5.3.

3.4 Procedure Outputs

3.4.1 Output Resulting from the Integration Process

An Integration Kit will be created as a result of applying the integration procedures to the inputs described in paragraph 3.3.1; the kit will be comprised of the following:

1) An Overview, consisting of a description of the purpose, viewpoint, context, assumptions, source documents, and conclusions made by the integrator.
2) **A Completed Subsystem Integration Matrix**

A Subsystem Integration Matrix form will be completed for the Subsystem®, as it integrates with the System®. This is an updated version of the form provided by the subsystem developer and shown in Figure 3. It is discussed further in paragraph 3.5.3.
3) **A Summary Version of the Matrix**

This is the same form as Item 2. It is marked to show groups of nodes which are analyzed together rather than individually. For example a single box might be analyzed without attention to the separate boxes composing the diagram which details the single box.

![Diagram](image)

4) **A copy of Subsystem Identifying Outside Arrows**

A copy of the Subsystem with all arrows which descend from arrows on $A$-$\emptyset$ or from tunnelled arrows are highlighted using wide arrows. Figure 3-5 illustrates a diagram from such a model.

5) **A Summary Model of the System Nodes Considered**

This is a standard IDEF model (lacking text and glossary) but consisting of FEO's (For Exposition Only) so that:

- less than 3 boxes are allowed on a diagram.
- extensive notes are provided and standard box numbers and ORE rules are waved to encourage notations explaining the structure of the model.
- highlighting of "external" arrows as seen in Figure 3-5 is used.

The procedure for producing this model is given in paragraph 3.5.2. Figure 3-6 shows a sample diagram from such a model.

This type of model is discussed further in paragraph 3.5.1.
Figure 3-5. Subsystem Identifying Outside Arrows

Figure 3-6. Standard IDEF0 Model
6) **Text Comparison of Nodes Considered**

For each node (or group of nodes) identified in Item 3 a discussion of the comparison of node titles, texts, glossaries and external arrows is developed. This part of the document, and the graphic material which may accompany it are discussed in paragraphs 3.5.4, 3.5.5 and 3.5.6.

7) **Exception Report**

The Exception Report contains an explanation for each numbered exception item which appears on the Integration Matrix or in the Text Comparison. The exceptions are numbered and described chronologically, using text plus copies of System@ or Subsystem@ diagrams as necessary to illustrate the exception item. Exceptions include arrow naming discrepancies, differences in glossary term usage and non-matching external arrow identification between related system and subsystem nodes.

The Exception Report may contain a Recommendations Section at the end of each Exception Report item. These recommendations may include:

- Recommendations for further System@ decomposition.
- Recommendations for additional System@ arrows, where Subsystem@ arrow attributes could not be found.
- Recommendations for modifications or corrections.

8) **An updated MFG@**

The integration team will provide new diagrams for all MFG@ diagrams which contain nodes supported by the new subsystem, or which are "parents", "grandparents" etc. of such diagrams. The supported boxes will carry a support arrow labeled subsys only or, for boxes with only one supporting subsystem, will read subsys/subsystem name. The team will maintain and deliver a node list type matrix (see Figure 3-7) to summarize the supporting subsystems. Each check in this matrix will represent one or more dots in the integration matrix of the subsystem indicated. By referring to that subsystem matrix, the reader can determine precisely which subsystem nodes support the CV node in which he is interested.
Figure 3-7. Architecture-Subsystem Integration Matrix
3.5 Integration of "AS IS" Subsystems

This section discusses the procedures for integration beginning with the delivery of the Integration Matrix by the subsystem developer who has completed an "AS IS" Subsystem. The section deals with the efforts of the integration team, a group which brings to the analysis an industry perspective of System and Subsystem.

3.5.1 Identification of Subsystem External Arrows

Since integration is concerned with the external interfaces of a subsystem, not with its inner workings, this phase of integration considers only 'hose arrows which terminate outside the subsystem. Examination of arrows which start and end within the subsystem occurs only during the more detailed examination which occurs during integration of the Subsystem "TO BE" model. The procedure, which is carried out by the integration team, begins by highlighting all arrows on Subsystem/A/. Arrows on A are then highlighted if they carry either an ICOM from one of the highlighted arrows or parentheses in place of an ICOM. This procedure is followed throughout the model until all diagrams have been considered. Figure 3-8 shows a parent diagram and the diagram which details one of its boxes.

The highlighting of arrows depends only on tunnelling and on highlighting of arrows on the parent. Discrepancies in arrow naming are noted for exception reports, (See paragraph 3.5.6) but are not otherwise traced. The use of this form of the model is discussed in paragraph 3.5.5.

3.5.2 Summing the Supported Nodes of System

In this step, the nodes of System which are supported by Subsystem are grouped into a coherent model. This grouping is performed bottom up. APPENDIX A shows a sample output of this procedure.
Figure 3-8. Parent Diagram with Detail
The summary is carried out for two reasons:

- To provide an overview of the topic in order to:
  - Provide top-down understanding
  - Highlight and focus on any pre-existing problems or discrepancies.

- To segregate 'internal' arrows and 'external' arrows for different treatment.

The procedure consists of the following steps:

1) All supported boxes are highlighted, and their interface arrows are adjusted to meet the changes noted by the subsystem developer.

2) On diagrams with more than one supported box, non-supported boxes are marked out (actually deleted for final reports). All arrows which do not touch supported boxes are marked out. Remaining arrows which touch non-supported boxes become external arrows. See Figure 3-9.

![Diagram of removal of unsupported box](image)

**Figure 3-9. Removal of Unsupported Box**
3) For diagrams with a single supported box, the same result is obtained by treating the arrows around the box as external arrows.

4) Parent diagrams of processed diagrams are treated in two ways:

a) Diagrams with a single child diagram which is supported are labeled as "channel diagrams." This indicates that the processing of the next higher diagram will be carried out by looking at the next lower diagram.

b) On diagrams with more than one supported child, the arrows around each parent box are relabeled, deleted, or added to until they match the external arrows of the child. The box which the above figure details would have controls labeled "b" and "d", inputs of "a" and "e" and outputs "c" and "f". The supported nodes at the next lower level (or even lower levels if reasonable) are listed under the parent box in place of the usual DRE. For Figure 3-9, nodes one and three would be listed on the parent. Boxes whose child diagrams have no supported nodes, and the arrows touching those boxes, are treated as were those in Step 2. This step often requires creation of a new diagram. The old diagram is marked "redrawn" and left as a documenting FEO of the redrawn diagram. Original box numbers are used. See Summary/A0 in APPENDIX A.
5) When a single diagram (A0) level is reached, the process reverses. In the top down pass, two steps are executed:

- New ICOM's are noted (or old ones confirmed) to check the process. In doing this, "channel" diagrams may be skipped.
- The arrows which are now external to the model (as discussed in paragraph 3.4.2) are highlighted.

6) The processed diagrams are assigned:

- a new model name
- a FEO number in addition to the basic node number.

Appendix A shows a working level in the preparation of such a model. The boxes supported are those across the top of Figure 3-10.

3.5.3 Maintenance and Summing of the Integration Matrix Form

The integration team, upon receipt of the Integration Matrix Form (see paragraph 3.4.1) will total the dots in all rows and columns. An exception occurs when no SystemO node can be found which matches a SubsystemO node, or if more than one such node is found. In these cases, a chronological Exception Number will be entered into the Exception Column (adjacent to the SubsystemO Node Number column) on the Integration Matrix form (right-most column). Paragraph 3.5.7 discusses the exception report further.

In addition, revisions of the Integration Matrix Form may result from the analysis conducted in paragraphs 3.5.4 and 3.5.5. These revisions are the responsibility of the Integration Team.

Finally, the summary SystemO model from paragraph 3.5.2 and examination of the Matrix itself will lead to decisions to group nodes for further study in paragraphs 3.5.4 and 3.5.5. This occurs where:

1) One SubsystemO node supports several related SystemO nodes.

2) One SystemO node is supported by several SubsystemO nodes.

3) A limited group of SubsystemO nodes support a limited group of SystemO nodes.
Figure 3-10 shows four conditions which may exist:

1) At Note 1, one box (Subsystem/All) supports 3 closely related boxes. The comparison needs to be made only between System/A3 and Subsystem/All. Individual consideration of System/A31, System/A32 and System/A33 is not required.

2) At Note 2, several subsystem boxes support a single box (System/All). In this case, the summing occurs in the Subsystem model; Subsystem/A3 is compared to System/All.

3) At Note 3, a limited group of Subsystem boxes support a limited group of System boxes. Such cases require individual analysis. Usually, summing of all boxes at each end is possible but the integration team may decide to sum over lesser groups or, occasionally, not at all.

4) At Note 4, a single box supports a single box and no summary need be made.

Figure 3-11 shows an integration matrix marked to show nodes which will be summed before comparison of supported and supporting nodes. The note marks on Figure 3-11 refer to the type numbers above. No occurrences of type 2 appear in the example. Note that some of the noted groupings do not meet the pure classifications given above.

The development of these groupings is guided by examining the output of paragraph 3.5.2. The groupings in turn guide considerations described in paragraphs 3.5.4 and 3.5.5. Several iterations of tentative groupings and revisions are to be expected.

3.5.4 Checking Text and Glossary

For each System0 node or group of nodes identified by executing paragraph 3.4.4, the integration team will prepare an Integration Textual Description, based upon the function model texts for each group of boxes.
**Figure 3-11. Annotated Intersection Matrix.**
The textual description will include a description of the exceptions if any, which are noted between the use of terms on the SystemO and SubsystemO diagrams and an Activity Analysis textual description of the activity differences (see Figure 3-12), including those functions that are included in the SystemO boxes which are not included in the SubsystemO boxes, and vice versa. These descriptions will be included, to present any similarity/difference noted by the integrator, not to elaborate or otherwise describe elements of SystemO or SubsystemO. Each exception will be classified as either critical, major or minor.

3.5.5 Comparison of External Arrows

For each SystemO node or group of nodes identified by executing Section 6.5 (TO BE REPLACED WITH CORRECT NUMBER), the integration team will prepare a comparison of external arrows. All external arrows reaching or leaving each group of nodes (SystemO and SubsystemO) are cross referenced.

Any SystemO arrows for which an acceptable match is not found are noted for exception reporting (see paragraph 3.5.6).

3.5.6 Exception Reporting

As each question or problem is encountered (see paragraphs 3.5.3; 3.5.4; 3.5.5 and 3.5.6), it is assigned a chronological number. The integration team maintains:

- A central file containing documentation for each exception.
- An index of all exceptions and an index of exceptions which the team considers open.
- A brief description of each open exception. These can be assembled at any time to provide a documentation of project status somewhat more extensive than that provided by the index of active exceptions.

3.5.7 Update of MFGO

The MFG model is annotated to show which boxes receive some level of support. To avoid clutter, the model diagrams do not reflect all supporting subsystems directly. However, any box which is supported itself or which has a "descendant" box which is supported, shows a support arrow (see Figure 3-13) marked "subsys". If only one subsystem supports the box, the subsystem is identified. The specific subsystems involved will be documented by an Architecture-Subsystem Integration Matrix (see Figure 3-11).
ACTIVITY DESCRIPTION
CV/A1242
SUB/A24113

EXCEPTIONS:

GLOSSARY
1. System Def. for ____:
   Subsystem Def. for ____:
2. ...

ACTIVITY ANALYSIS
Similarities:
Differences:

Figure 3-12. Integration Textual Description

Figure 3-13. Supported Box
When one subsystem reports that it supports a lowest level node supported by one or more other subsystems, the integration team will examine the interface between the subsystems. If the subsystems are clearly compatible, as when an output of one is an input to the other, no further action is required.

If interfaces between the subsystems are not clear, an exception will be added to the exception list for each of the subsystems.

3.6 Integrating a Subsystem Model Containing Generic Functions

The process of integrating a Subsystem with a split into multiple generic functions is analogous to the previous Section description except that it includes a preliminary step of creating a "summary FEO" of all generic functions, and then using the FEO for integration of the Subsystem with the System.

Preparing and Using a Summary FEO

For each group of generic functions in the Subsystem, create an IDEF0 FEO diagram which summarizes the activities portrayed by all generic functions in the group. Use the boxes of this FEO as the lowest-level Subsystem Nodes on the Matrix form, instead of the actual Subsystem nodes. Reference the node numbers with an "F" preceding the FEO box number, to indicate a generic function reference (e.g., "A123F1").

To be precise about each generic function integration, an Integration Matrix will be developed for each generic function to show how the function integrates with the FEO, and the FEO boxes supported by the generic function will be annotated with support arrows. In other words, the basic integration procedure will be applied to the FEO and generic function-models just as they are applied to the System and the Subsystem. Thus, the main integration effort shows generic integration, with specific details traceable via the FEO's second level of detail.

In other cases, a subsystem may perform one or more generic functions which are widely used. For example, a Group Technology system may provide for recovery of a part given its group characteristics. This ability might be useful at many points in manufacturing. In this case, a single summary box (or several, one for each of several functions) will be checked at many points in System.
4.1 **Introduction**

As part of the integration of subsystems into the Manufacturing Architecture (Subsystem into MFG0), a more complete form of arrow definition known as an "arrow trace" was developed and applied to MFG0. This new procedure incorporates the formerly developed glossary definitions of arrow labels, adding the following additional information to the textual definitions:

- A list of synonymous terms used for the data carried on the arrow.
- A list of source functions which generate the data carried on the arrow.
- A list of target functions which utilize the data carried on the arrow.
- A list of the sub-parts (origin components) comprising the data carried on the arrow, as shown by the arrow branching and joining structure.
- The name of the more inclusive data item or items (usage components) which contain the data carried on the arrow, as shown by the arrow branching and joining structure.

This more complete set of information has been deemed necessary in applying the integration procedure in order to document the complete impact of a subsystem on the total Manufacturing Architecture. Also, the arrow tracing procedure has been found to be helpful in pointing out modeling errors and inconsistencies in the arrow structures, such as inconsistent use of arrow labels.

**NOTE:** This procedure has been written with the assumption that the reader is familiar with the IDEF1 methodology.

4.2 **The General Arrow Trace Procedure**

In general, the Arrow Trace procedure traces the path of each arrow in the model to find the origin(s) and target(s) of an arrow. Before proceeding, a familiarity of the terminology used in the context of "tracing arrows" is required.

**Arrow** - A directed line segment having a specific label.

**Origin** - A function (box) which creates a specific arrow and/or the point at which a specific arrow first appears in a model.
Target - A function in which an arrow finally enters and/or the point at which an arrow label changes.

Origin Component - Arrows which join together to make-up the arrow being traced and/or an arrow whose name changed to that of the arrow being traced.

Usage Component - An arrow which is the result of the traced arrow's name change.

ICOM Code - The code that corresponds to the origin of the arrow.

The arrow trace begins by selecting an arrow. The person performing the arrow trace (the tracer) traces the origin(s), then the target(s).

The diagram showing the arrow being traced is then examined. Each time the arrow is shown entering or leaving a function (box), that function is examined to see if it has a decomposition diagram. If not, the function is listed as a source (leaving) or target (entering). If a decomposition diagram exists, the decomposition diagram is examined to find the continuation of the arrow path, and the trace continues.

If the trace procedure encounters an arrow which is open-ended (it does not enter or leave a box on the diagram being examined), the ICOM code of this "boundary arrow" is used to locate the arrow on the "parent" diagram, and the trace continues.

The most complex interaction occurs when an arrow is shown branching or joining. If the branches are not labeled differently from the main arrow path, this indicates multiple sources or targets. If the branches are labeled differently from the main arrow path, this indicates that the main data item is comprised of the sub-parts shown in the branches. In either case, the trace must continue until all sources or targets are identified.

4.3 The Detailed Trace

The above "general procedure" introduces the major arrow tracing concepts. The detailed procedure, described below and depicted in a flow diagram (Figure 4-1), presents the complete procedure, showing all possible situations encountered and the steps to be performed in each situation. The Arrow Trace Form is included as Figure 4-2 along with several examples (Figures 4-14, 4-15, 4-16, and 4-18) of completed forms resulting from the MFGO tracing effort.
Figure 4-1. Arrow Trace Procedure
At any instant, the goal of the trace is either to locate the origin(s) and target(s) of the selected arrow. Note that, since a single arrow may show multiple origins and targets, using the branch/join arrow structures of IDEF (see Figure 4-3), this will require a forward and backward trace of each branch to complete the arrow trace on a single arrow.

![Diagram of an arrow trace](image)

**Figure 4-3. Example of Multiple Sources and Targets of an Arrow**

For example, if arrow "A" (Figure 4-3) were being traced, Box 2 would be an origin and the origin trace would have to be continued in the decomposition of Box 1. Similarly, Box 5 would be a target and the target trace would have to be continued in the decomposition of Boxes 3 and 4. A method of keeping track of the branch being traced and the branches to be traced is left to the person performing the arrow trace.

The arrow trace begins by selecting an arrow. The procedure continues by tracing (1) origin(s), and (2) target(s) of the arrow. In each trace, at least one of four primary cases will be encountered (see Section 4.3.1) and are listed below.
1. Arrow has a label change.
2. Trace leads to a box.
3. Arrow is tunneled.
4. Arrow is a "Boundary Arrow."

Each case requires a decision and an appropriate action. These will be discussed in detail. Notice the branch/join does not require a decision. Each branch must be traced to complete the procedure for an arrow.

4.3.1 Trace Origin (Figure 4.1, Box 2)

To trace the origin(s) of an arrow, the selected arrow should be followed in the direction of the tail, i.e., in a backward direction. Following this course, at least one of the cases listed above will be encountered. Each case is considered in detail below.

4.3.1.1 Case I: Label Change (Figure 4-1, Box 2.2)

When a trace leads to a label change, the tracer must decide if the new label is appropriate. If the new label is appropriate, then the change must be documented on the Arrow Trace Form (Figure 4-2). This documentation requires the following steps. (Figure 4-1, Boxes 2.4 and 2.5)

1. Record ICOM Code and name of the new arrow as an "origin component" on the Arrow Trace Form.
2. Record the ICOM code of the arrow being traced (subject arrow) at the first occurrence of the arrow; i.e., that point at which the label changed.

In Figure 4-4, the arrow labeled "C" is the subject arrow. Notice that in tracing "C" backward, the label changes to "A" and "B." If these are appropriate changes, then the output arrows, "A" and "B," are listed as "origin components." The ICOM code of the control arrow "C" is then recorded on the Arrow Trace Form.

If the new label is not appropriate, then the tracer must recommend a revision to the model. A list of recommended revisions should be compiled containing reference to position in the model and explanations of the changes.

When the recommended change is that a different new label be used, the tracer continues the documentation as described by the previous example.
Figure 4-4. Example of Case 1: Label Change, Appropriate

When the recommendation is that there be no label change, then the tracer only needs to continue the trace.

In Figure 4-5, arrow "C" is the subject arrow. The label change from "C" to "B" is deemed inappropriate. The decision is that the label should be "D" rather than "B." This label change is then documented in the same manner as described in steps 1 and 2 of this section.

The label change from "C" to "A" in Figure 4-5 may also be inappropriate, i.e., the label should remain "C." In this case, the recommendation should be documented and the procedure continued from Box 2.1 of Figure 4-1.

4.3.1.2 Case 2: Trace Leads to a Box (Figure 4-1, Box 2.9)

When the arrow trace leads to a box, the box must be examined for decomposition. If the box decomposes, the tracer simply continues the trace in the child diagram. If the box does not decompose, then that box is considered an origin. The ICOM code of the arrow, at the point of contact with the box, is recorded under ICOM code on the Arrow Trace Form.
Figure 4-5. Example of Case 1: Label Change, Inappropriate

In Figure 4-6, "A" is the subject arrow. Tracing "A" backward leads to box one. If box one decomposes, the tracer locates the child diagram of box one and continues the procedure in that diagram. If box one does not decompose, then it is considered an origin of arrow "A" and the ICOM code of "A" at the point of contact with box one is entered under ICOM code on the Arrow Trace Form.

4.3.1.3 Case 3: Tunneled Arrow (Figure 4-1, Box 2.13)

When the subject arrow is tunneled, the ICOM code is obtained at the point the arrow first appears in the model.

In Figure 4-7, arrows "A" and "B" would not be referenced in the parent or child diagrams respectively. The ICOM codes for each arrow would be taken from their points of contact with box one.

4.3.1.4 Case 4. Boundary Arrow (Figure 4-1, Box 2.14)

If the subject arrow is a boundary arrow, then the tracer must continue the trace in the parent diagram.
If this box decomposes, continue trace in child.

If box 1 does not decompose, it is an origin.

Figure 4-6. Example of Case 2: Trace Leads to a Box

Figure 4-7. Example of Case 3: Tunneled Arrow

4.3.1.5 Error in Trace (Figure 4-1, Box 2.16)

If tracing an arrow does not lead to any of the four cases described, then the tracer has made an error and should begin the trace over.

4.4.1 Trace Target (Figure 4-1, Box 3)

The logic and procedure for tracing the target(s) of an arrow are basically the same. A major portion of the differences is in the documentation. Only the differences will be addressed in the remaining sections of this procedure.
4.4.1.1 Case 1: Label Change (Figure 4-1, Box 3.2)

Decisions in this case are identical to those in Section 4.3.1.1. Differences in the procedure occur in the documentation required when the label change is appropriate. The ICOM code of the new arrow is recorded as a "target" on the Arrow Trace Form. The label change is then referenced by adding a footnote (parenthesized digit) to the target ICOM code. The new label is then documented using the corresponding footnote number under "usage component" on the Arrow Trace Form. An example of this documentation is presented in Figure 4-8 below.

![Diagram](image)

Figure 4-8. Target Trace Label Change Documentation

The figure above depicts an "A1" diagram in which arrow "A" is the subject arrow. The arrow's label appropriately changes to "B" and "C." Arrows "B" and "C" must then be identified as "usage components" on the Arrow Trace Form. This is done by listing their footnoted ICOM codes under "targets" and their labels with corresponding footnote numbers under "usage components."
4.4.1.2 Case 2: Trace Leads to Box (Figure 4-1, Box 3.10)

If the box decomposes, the trace is continued in the child diagram. If the box does not decompose, then the ICOM code of the arrow at the entry point to the box is listed as a "target" on the Arrow Trace Form (see Figure 4-9).

![Diagram of Case 2: Trace Leads to Box]

Figure 4-9. Example: Arrow Trace (Target) Leads to Box

4.4.1.3 Tunneled Arrow (Figure 4-1, Box 3.14)

If the arrow is tunneled at the entry to a box, the ICOM code of the arrow at that point is listed as a "target." Notice it does not matter if the box decomposes since the tunneled arrow will not appear on the child diagram.

If the arrow is tunneled at the boundary of the diagram, then no action is required. The arrow has no target at that point.

4.4.1.4 Boundary Arrow (Figure 4-1, Box 3.15)

If the subject arrow is a boundary, then the tracer must continue the trace in the parent diagram.

4.5 Examples from MFGØ Arrow Trace

This section presents selected examples from the MFGØ Arrow Trace. References will be made to previous sections of this procedure and the appropriate following figures in an effort to provide practical examples of this procedure.
4.5.1 **Label Changes (reference paragraphs 4.3.1.1 and 4.4.1.1)**

The trace numbers used in this example are from the MFG0 model and are intended to identify specific arrows on specific diagrams. The format key for these trace numbers is interpreted as follows:

Axxx.BAN

Axxx - identifies node in MFG0
B - identifies the Box on that diagram
A - identifies ICOM Arrow type
N - identifies the relative Number of the arrow.

Figure 4-10 is a sample diagram from MFG0. In tracing the origin of the arrow labeled "job sequence analysis," notice that the label changes to "production sequence." Figure 4-13 shows "A132.102 production sequence" as an origin component and the ICOM Code "A132.211" listed for "job sequence analysis."

Tracing the target labeled "production sequence" in Figure 4-10 also leads to the label change to "job sequence analysis." Figure 4-12 shows "A132.211" listed as a target (with footnote) and the new label (footnoted), "job sequence analysis," under usage component.

4.5.2 **Trace Leads to a Box (reference paragraphs 4.3.1.2 and 4.4.1.2)**

The arrow labeled "kit plan" in Figure 4-10 is an example of an arrow trace (both target and origin) which leads to a box. Tracing the origin of "kit plan" yields the ICOM code "A132.202" as shown in Figure 4-11. Tracing the origin yields "A132.311" as a target.

4.5.3 **Tunneled Arrow (reference paragraphs 4.3.1.3 and 4.4.1.3)**

In Figure 4-14, the arrow labeled "WBS & drawing tree" is an example of a tunneled arrow. The appropriate documentation is shown on Figure 4-15.

4.5.4 **Boundary Arrow (reference paragraphs 4.3.1.4 and 4.4.1.4)**

Arrows labeled "line assembly and installation plan" and "resource plans" on Figure 4-10, are examples of boundary arrows.
<table>
<thead>
<tr>
<th>COMPONENT(S)</th>
<th>USES COMPONENT(S)</th>
<th>(1) Job sequence analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>A132.211</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A132.312</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NODE:</td>
<td>TITLE:</td>
<td>NUMBER:</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>A132.211</td>
<td>JOB SEQUENCE ANALYSIS</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4-13. Job Sequence Analysis**
SECTION 5
IDEF1 INTEGRATION PROCEDURE

5.1 Introduction

This procedure is designed to serve as a reference guide for the combining of two or more IDEF1 information models into a single information model. The concepts used to facilitate the combining of IDEF1 information models are described and depicted in the various examples throughout this manual. This procedure is designed to be a working reference for the experienced information modeler.

This procedure assumes that the integration modeler has a working knowledge of IDEF1 information modeling methodology and has experience in building multiple IDEF1 information models.

The procedure is based on two assumptions regarding the quality of the models to be used in the integration process. These assumptions are: 1) the models correctly apply the IDEF1 methodologies, and 2) the models accurately reflect the factory views they represent. The quality of the source models will have an impact on the ease with which the models can be integrated. Models which do not correctly apply the IDEF1 methodology or do not accurately reflect the environments they represent can cause the resulting integrated model to lack credibility.

The modeler must also guard against any inadvertent changes to the views of the source models, as a result of the integration process. This can occur rather easily and the modeler should refer to the source models frequently during the integration process to ensure that the integrated model maintains the source model views.

The modeling team should consist of modelers and reviewers who represent the various source models. A team established in this way will provide additional guarantees that the source model views are maintained in the integrated model.

In the course of integrating IDEF1 information models, the modeler may find that, between the source models being integrated, there exist no common entity classes. As a result, "bridges" will have to be built between the models and therefore, new entity classes will result.

New entity classes may also be created from resolutions of discrepancies that arise as a result of the varying views of the models being integrated. The resolution of these discrepancies will be dealt with in the sections that follow.
Any number of IDEF1 information models can be integrated using this procedure. However, the more models being integrated, the more involved the record keeping becomes to provide traceability back to the source models.

This procedure utilizes a five phase approach to the development of an integrated model. This approach is consistent with the five phase development of an IDEF1 information model. The documentation produced by this procedure also parallels the IDEF1 information modeling methodology. The differences, due to the nature of the integration process, will become evident from the discussion that follows. The five phases for developing an integrated model are as follows:

**Phase Zero**

Phase Zero documents the context of the integrated model. In this phase, the scope of the integrated model is defined, its objectives are stated, and the source data identified.

**Phase One**

In Phase One, the objective is to identify and define the candidate entity classes to be used in the integrated modeling effort.

**Phase Two**

In Phase Two, the initial relation classes between the candidate entity classes will be identified.

**Phase Three**

In Phase Three, the key classes for each of the entity classes in the integrated model will be identified and defined.

**Phase Four**

In Phase Four, the integrated model will be populated with its non-key attribute classes.

The result of the integration process will be a new model which will reflect the combined views of all of the source models. It is of utmost importance that the integrated model accurately represent the views of the various source models and that the components of the source models are able to be identified within the context of the completed integrated model. Maintaining this approach will ensure maximum usability of the model to the enterprise.
5.2 Phase Zero

The integrated information model must be described and defined in terms of both its ambitions and its limitations. This will be accomplished through a statement of the strategic objective and definition of the scope of the integrated model. The strategic objective consists of two elements. These elements are a statement of purpose and a statement of viewpoint. It is likely that the models to be integrated will have been developed from different viewpoints with differing strategic objectives. For the resulting integrated model to be meaningful, a strategic objective must be synthesized that will accurately reflect the strategic objectives of the source models without changing their intent. This "synthesis of viewpoint" will be evident throughout this procedure and is an integral part of the integration process. An example of a synthesized strategic objective for an integrated model is provided in Figure 5-1.

---

**Figure 5-1 Strategic Objective**
The scope of the source models will likewise have been developed to satisfy a specific factory view. A scope must be established for the integrated model that will satisfy the scopes of the various source models. One way to establish the scope of the integrated model is to view the problem domain of the source models from an IDEFO perspective. An analysis of this IDEFO perspective will then help clearly establish the scope and context of the integrated model. An example of an IDEFO integrated model scope viewed from the IDEFO perspective is provided in Figure 5-2.

To provide traceability of data used in the integration process, a source material log (SML) and a source data list (SDL) are constructed. Source material in this context will be the various source models which are to be integrated. The source material log lists all the source models used to create the integrated model. An example of this source material log is provided in Figure 5-3.

Figure 5-2. IPS Composite View Scope
<table>
<thead>
<tr>
<th>Source Material</th>
<th>Source Material Name/Description</th>
<th>Received From</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM-1</td>
<td>Factory View-Plan for Manufacture</td>
<td>Northrop-G. C. Willoughby</td>
<td>Kit Dated 7/15/81</td>
</tr>
<tr>
<td>SM-2</td>
<td>Factory View-Make and Administer Schedules and Budgets</td>
<td>Rockwell-131 House</td>
<td>Kit Dated 7/11/81</td>
</tr>
<tr>
<td>SM-3</td>
<td>Factory View-Plan Production</td>
<td>G. D. F. W. Smith</td>
<td>Kit Dated 7/8/81</td>
</tr>
</tbody>
</table>

Figure 5-3. Source Material Log

The source data list (SOL) is constructed by listing all of the entity class and attribute class names from the source models. (Figure 5-4) Each item on the source data list is given a unique identification to provide traceability to its originating source model. An attempt is made during this listing process to identify commonality or redundancy of source data names which have common meanings. This information will be used during Phase One and Phase Three to construct integrated model entity classes and attribute classes respectively.

The last step in Phase Zero is the identification of author conventions. The use of author conventions is intended to enhance the presentation of the material and facilitate a better understanding and appreciation of the integrated model. They are not formal extensions of the modeling technique nor violations of the modeling technique (Ref. IDEFL Manual).

As appropriate, during the integration process, Phase Zero Kits are prepared for distribution and review. A Phase Zero Kit is composed of a cover sheet followed by some number of pages representing one or more sections of Phase Zero documentation. Phase Zero Kits consist mostly of textual material and should average no more than 50-75 pages in length. Each kit should require no more than one or two hours for review.

5-5
To summarize, the objective of Phase Zero is a clearly established set of products which include:

**Strategic Objective Definition**
- Purpose
- Viewpoint
- Context

**Scope and Viewpoint**

**Source Material Log (SML)**

**Source Data List (SDL)**

**Author Conventions**

### 5.3 Phase One

The objective of Phase One is to identify and define the entity classes from the source models that will be included in the integrated model. These candidate entity classes are drawn from the Source Data List constructed in Phase Zero. It is during this identification process that the issue of source model entity class commonality or redundancy, partially identified in Phase Zero, is addressed.

---

**Figure 5-4. Source Data List**
The process of identifying the entity classes for inclusion in the integrated model is as follows: One of the source models to be integrated is chosen as a baseline model. The selection of the baseline model is strictly arbitrary. Its purpose is to provide a starting point. Each of the entity classes in the remaining models is compared with the baseline model. Where an identical or similar entity class definition exists between the models, the affected entity classes become candidates for combining into a single integrated entity class. The key point is commonality or similarity (defined as "commonality of intent") of the definitions. Commonality or similarity of entity class names alone may be misleading because of differing source model viewpoints and differing factory view usage of terms (i.e., two factory views using the same term but with different definitions). This commonality of entity class definitions from the source models represents "overlap" between the models.

For each group of common or "overlap" entity classes, a single entity class name and definition is synthesized which most accurately reflects the viewpoint and strategic objective of the integrated model. The resulting entity class name may not be identical to any of the source model entity class names from which it originated, but it must reflect the meaning of the originating source entity classes. An example of the entity class name and definition synthesis is provided in Figure 5-5.

Figure 5-5. Integrated Entity Class Development
A glossary page is prepared for each entity class synthesized. The entity class names which were not used for the integrated entity class name, may be used, where appropriate, as synonyms for the integrated entity class name and are also listed on the glossary page. (Figure 5-6)

The entity classes for which no commonality existed are now examined for relevancy to the viewpoint and strategic objective of the integrated model. Glossary pages are prepared for each of the remaining entity classes identified as being within the scope of the integrated model. Those candidate entity classes falling outside the integrated model scope are eliminated from the integrated model. The eliminated entity classes are listed on a text page, along with their source data list (SDL) identifiers, and a statement explaining the reason(s) for non-inclusion in the integrated model. (Figure 5-7).

The candidate entity classes which survive the above process form the Entity Class Pool for the integrated model. The surviving entity classes are recorded on the Entity Class Pool Form (Figure 5-8), and new entity class numbers assigned.

<table>
<thead>
<tr>
<th>ENTITY CLASS NAME:</th>
<th>Drawing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTITY CLASS LABEL:</td>
<td>Drawing</td>
</tr>
<tr>
<td>ENTITY CLASS DEFINITION:</td>
<td>A graphical representation of an object which reflects its geometric configuration, dimensions and construction (form, fit and function).</td>
</tr>
</tbody>
</table>

ENTITY CLASS SYNOMYS:
- Blueprint
- Engineering Drawing
- Brownline

Figure 5-6. Entity Class Definition: Drawing
Figure 5-7. Entity Class Not Used in Integrated View

Figure 5-8. Entity Class Pool
An entity class source model cross reference is now prepared. This cross reference provides traceability for each integrated entity class to its originating source model(s). An example of this source model entity class cross reference is provided in Figure 5-9.

At appropriate points during Phase One, kits are structured and circulated for review and comment.

A Phase One Kit will typically consist of the following:

- Kit Cover Sheet
- Entity Class Pool
- Source Model Entity Class Cross Reference
- 15-20 New Entity Class Definition Pages Per Kit

Review time for each kit should not exceed one or two hours.

<table>
<thead>
<tr>
<th>E.C. NO.</th>
<th>ENTITY CLASS NAME/COMMENT</th>
<th>E.C. NO.</th>
<th>ENTITY CLASS NAME</th>
<th>E.C. NO.</th>
<th>ENTITY CLASS NAME</th>
<th>E.C. NO.</th>
<th>ENTITY CLASS NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1C</td>
<td>Section Activity</td>
<td>E501</td>
<td>Section Activity</td>
<td>E33</td>
<td>Material</td>
<td>E49</td>
<td>Raw Material</td>
</tr>
<tr>
<td>E2C</td>
<td>Configured Item</td>
<td>E519</td>
<td>Configured Item</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3C</td>
<td>Material</td>
<td>E505</td>
<td>Material</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4C</td>
<td>* * * Not used * * *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5C</td>
<td>Statement of Work</td>
<td>E533</td>
<td>Statement of Work</td>
<td>E21</td>
<td>Contract</td>
<td>E38</td>
<td>Contract</td>
</tr>
<tr>
<td>E6C</td>
<td>Sales Contract</td>
<td>E520</td>
<td>Contract</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(&quot;Sales&quot; added to distinguish from Purchase Contract)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E7C</td>
<td>Engineering Change Request</td>
<td>E521</td>
<td>Drawing Change Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E8C</td>
<td>Engineering Change Proposal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E9C</td>
<td>Engineering Order</td>
<td>E32</td>
<td>Engineering Order</td>
<td>E21</td>
<td>Engineering Release</td>
<td>E23</td>
<td>Engineering Change Notice</td>
</tr>
<tr>
<td>E10C</td>
<td>Approved Engineering Order</td>
<td>E71</td>
<td>Approved Engineering Order</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E11C</td>
<td>Drawing</td>
<td>E525</td>
<td>Drawing</td>
<td>E36</td>
<td>Drawing</td>
<td>E20</td>
<td>Engineering Drawing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Part</td>
</tr>
<tr>
<td>E12C</td>
<td>Part—(By definition an Assembly could be a part)</td>
<td>E536</td>
<td>Part</td>
<td>E28</td>
<td>Part Type</td>
<td>E1</td>
<td>Part</td>
</tr>
<tr>
<td>E13C</td>
<td>Component Part</td>
<td>E552</td>
<td>Assembly</td>
<td>E550</td>
<td>Component Part</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E14C</td>
<td>Section</td>
<td>E544</td>
<td>Section</td>
<td>E6</td>
<td>Work Breakdown Structure Item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E15C</td>
<td>Work Breakdown Structure Item</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-9. Entity Class Name Cross Reference
In summary, Phase One produces the following products for the integrated model:

- Entity Class Pool
- Entity Class Definitions
- Entity Class Source Model Cross Reference
- Phase One Kit(s)

5.4 Phase Two

The objective of phase two in integrated model development is the identification and definition of relation classes and relation class labels. The relation classes to be used in the integrated model are those in the source models which apply to the integrated entity classes identified in phase one. Rough (pencil) drafts of entity class diagrams (with an entity class box only) are constructed for each entity class in the integrated model. (Figure 5-10) The relation classes, along with their respective labels, from each source model entity class is applied to its integrated entity class counterpart. The rough draft integrated entity class diagrams are updated to reflect the relation classes and labels represented in the source models. (Figure 5-11)
In the "overlap" area of the integrated model, conflicting relation class syntax and/or conflicting relation class labels (meaning) may occur because of the differing viewpoints of the source models. FEOs (For Exposition Only) (called refinement alternative FEOs) are constructed that offer alternatives to the conflicting views (Figure 5-12). Actual resolution of these conflicting viewpoints will be accomplished later on in Phase Three.

Triads (three entity classes related directly to each other) may also occur due to differing viewpoints of the source models. FEOs are constructed to illustrate "triads" that result from the integrated process, along with suggested refinements. (Figure 5-13) Triads are resolved through the Phase Two kit review cycle.

Glossary pages are prepared for those relation class definitions, appearing in the source models, that are appropriate to the integrated model. New relation class definitions resulting from the integration process are also incorporated into the integrated model and documented on glossary pages as appropriate.
Figure 5-12. Refinement Alternative FEO: Conflicting Syntax

Figure 5-13. Triad "FEO"
The entity class diagrams are now formalized and the entity class node cross reference constructed. (Figure 5-14). The entity class node cross reference provides an easily usable index to the relation classes contained in the integrated model.

Source model "views" (projections) from the integrated model can now be constructed. These source model views (Figure 5-15) allow each source model to be seen in the context of the integrated model. A source model view is constructed by replacing each source model entity class with its integrated entity class. Any changes to relation class(es) and labels are also shown. These source model views help to validate the structure and semantic intent of the integrated model.

At appropriate points in Phase Two, kits are prepared for review and comment. A typical Phase Two kit should contain from thirty to fifty pages of new material. It should require no more than one or two hours for review. A Phase Two kit will consist of the following:

- Kit Cover Sheet
- Source Model Entity Class Cross Reference
- Source Model Views

![Figure 5-14. Related Entity Class Node Cross Reference](image)
To summarize, the objective of Phase Two is to produce the following products.

- Source Model Views (of the integrated model)
- Entity Class Diagrams
- Entity Class Node Cross Reference
5.5 Phase Three

In Phase Three, the Attribute Class Pool is established, key attribute class(es) are assigned to each entity class in the integrated model, and key class migration occurs.

Using the previously selected baseline model as a starting point, the attribute class definitions of the source models are compared with the baseline model for commonality. Where an identical or similar attribute class definition exists between the models, the affected attribute classes are candidates for combining into a single integrated attribute class. The key point is commonality or similarity (defined as "commonality of intent") of attribute class definition. Commonality or similarity of attribute class names may be misleading because of differing source model viewpoints and differing factory view usage of terms (i.e., two factory view using the same term, but with different definitions).

For each group of "common" attribute class definitions, a single attribute class name and definition is synthesized for the integrated model. The resulting attribute class name may not be identical to any of the source model attribute class names from which it originated, but it must reflect the meaning and intent of the source attribute classes. An example of an integrated attribute class name and definition synthesis is provided in Figure 5-16. A glossary page is prepared for each attribute class synthesized. (Figure 5-17)

Attribute class names which were not used for integrated attribute class name(s) may be used, where appropriate, as synonyms for the integrated attribute class name. These synonyms are listed on the glossary page.

The remaining attribute classes are examined for relevancy to the viewpoint and strategic objective of the integrated model, and their applicability to the integrated entity classes. Those candidate attribute classes which fall outside the scope of the integrated model (based in part on the eliminated entity classes from Phase One) are eliminated. The eliminated attribute classes are listed on a text page (with their source data list [SDL] identifier) stating the reasons for non-inclusion in the integrated model. (Figure 5-18). The candidate attribute classes which survive, together with the synthesized attribute classes form the Attribute Class Pool, and are recorded on the Attribute Class Pool Form (Figure 5-19). A new attribute class number is assigned to each member of the Integrated Model Attribute Class Pool.
Figure 5-16. Integrated Attribute Class Development

Figure 5-17. Attribute Class Definitions: Engineering Change Request
<table>
<thead>
<tr>
<th>A.C. NO</th>
<th>ATTRIBUTE CLASS NAME</th>
<th>SOURCE</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15</td>
<td>Building Number</td>
<td>RI</td>
<td>Outside Scope of Integrated Model</td>
</tr>
<tr>
<td>A64</td>
<td>Building Location</td>
<td>RI</td>
<td>Outside Scope of Integrated Model</td>
</tr>
<tr>
<td>53</td>
<td>Tool Order Number</td>
<td>SO/FW</td>
<td>Duplicate</td>
</tr>
<tr>
<td>A4</td>
<td>Shift Number</td>
<td>RI</td>
<td>Outside Scope of Integrated Model</td>
</tr>
</tbody>
</table>

Figure 5-18. Attribute Classes Not Used In Integrated Model

<table>
<thead>
<tr>
<th>A.C. NO</th>
<th>ATTRIBUTE CLASS NAME</th>
<th>SOURCE DATA</th>
<th>I.D. NO</th>
<th>ATTRIBUTE CLASS NAME</th>
<th>SOURCE DATA</th>
<th>I.D. NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-1</td>
<td>Material Specification</td>
<td>SD-10</td>
<td>-</td>
<td>A-18</td>
<td>SD-60</td>
<td>-</td>
</tr>
<tr>
<td>A-2</td>
<td>Contract Number</td>
<td>SD-42</td>
<td>-</td>
<td>A-19</td>
<td>SD-200</td>
<td>-</td>
</tr>
<tr>
<td>A-3</td>
<td>Engineering Change Request Number</td>
<td>SD-54</td>
<td>-</td>
<td>A-20</td>
<td>SD-101</td>
<td>-</td>
</tr>
<tr>
<td>A-4</td>
<td>Engineering Change Proposal Number</td>
<td>SD-62</td>
<td>-</td>
<td>A-21</td>
<td>SD-104</td>
<td>-</td>
</tr>
<tr>
<td>A-5</td>
<td>Drawing Number</td>
<td>SD-67</td>
<td>-</td>
<td>A-22</td>
<td>SD-108</td>
<td>-</td>
</tr>
<tr>
<td>A-6</td>
<td>Part Number</td>
<td>SD-80</td>
<td>-</td>
<td>A-23</td>
<td>SD-155</td>
<td>-</td>
</tr>
<tr>
<td>A-7</td>
<td>Configuration Baseline Number</td>
<td>SD-121</td>
<td>-</td>
<td>A-24</td>
<td>SD-235</td>
<td>-</td>
</tr>
<tr>
<td>A-8</td>
<td>End Item Serial Number</td>
<td>SD-128</td>
<td>-</td>
<td>A-25</td>
<td>SD-179</td>
<td>-</td>
</tr>
<tr>
<td>A-9</td>
<td>Line Number</td>
<td>SD-125</td>
<td>-</td>
<td>A-26</td>
<td>SD-190</td>
<td>-</td>
</tr>
<tr>
<td>A-10</td>
<td>Process Plan Number</td>
<td>SD-164</td>
<td>-</td>
<td>A-27</td>
<td>SD-182</td>
<td>-</td>
</tr>
<tr>
<td>A-11</td>
<td>Milestone Description</td>
<td>SD-186</td>
<td>-</td>
<td>A-28</td>
<td>SD-126</td>
<td>-</td>
</tr>
<tr>
<td>A-12</td>
<td>Contract Tool Number</td>
<td>SD-177</td>
<td>-</td>
<td>A-29</td>
<td>SD-217</td>
<td>-</td>
</tr>
<tr>
<td>A-13</td>
<td>Equipment Description</td>
<td>SD-200</td>
<td>-</td>
<td>A-30</td>
<td>SD-233</td>
<td>-</td>
</tr>
<tr>
<td>A-14</td>
<td>Department Number</td>
<td>SD-101</td>
<td>-</td>
<td>A-31</td>
<td>SD-231</td>
<td>-</td>
</tr>
<tr>
<td>A-15</td>
<td>Section Number</td>
<td>SD-195</td>
<td>-</td>
<td>A-32</td>
<td>SD-256</td>
<td>-</td>
</tr>
<tr>
<td>A-16</td>
<td>Activity Name</td>
<td>SD-186</td>
<td>-</td>
<td>A-33</td>
<td>SD-259</td>
<td>-</td>
</tr>
<tr>
<td>A-17</td>
<td>Change Number</td>
<td>SD-215</td>
<td>-</td>
<td>A-34</td>
<td>SD-271</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 5-19. Attribute Class Pool
The attribute class source model cross reference is now prepared. This cross reference provides traceability for each integrated attribute class back to its originating attribute class(es) in the source models. An example of an attribute class source model cross reference is provided in Figure 5-20.

The next activity in Phase Three is the assignment of key classes to each entity class in the integrated model. Using the source models as a guide, assign key classes to the integrated entity classes which: A) are fully independent, and B) are not an "overlap" or synthesized entity class. The entity classes which are "overlap" entity classes will have their key classes determined during key class migration.

Key classes are assigned to each of the non-"overlap" integrated entity classes by comparing the attribute classes used as key class(es) for the source model entity class, to the attribute class(es) in the integrated Attribute Class Pool. The integrated attribute class(es) which correspond to the source attribute class(es) are selected as the key class(es) of the integrated entity class.

![Figure 5-20. Attribute Class Name Cross Reference](image-url)
As the key classes are assigned, they are noted on pencil draft attribute class diagrams. (Figure 5-21) All key class members must pass the "no null," "no repeat criteria." (Ref. IDEF1 manual) Those attribute classes which fail the "no null" and "no repeat" criteria typically result in the creation of new entity classes to satisfy the IDEF1 methodology requirement of unique identification. The "new" entity classes which result are added to the Entity Class Pool, entity class numbers assigned, and glossary pages (definitions) are prepared.

The next step is key class migration. One role the migration of key classes serves is to validate the assigned relation classes. Before key class migration can begin, conflicting syntax identified in Phase Two is resolved. The modeler(s) should choose the syntax which best satisfies the intent of the integrated model. The result of this decision is documented by FEOs for review during the Phase Three kit review cycle. Multiple (non-synonymous) relation classes are left in place at this stage. These will be resolved through the key class migration process.

Key class migration is initiated from the fully independent, non-overlapped integrated entity classes, and progresses to the other integrated entity classes in accordance with IDEF1 methodology.

Figure 5-21. Attribute Class Diagram: Department
At some point in the key class migration process, the modeler will be confronted with an "overlap" entity class. In all likelihood, for these 'overlap' entity classes the key class structure(s) from the source model(s) will be different in each model, so the appropriate integrated key class(es) for these "overlap" integrated entity classes will have to be synthesized. The key class assignment process for these 'overlap' entity classes is as follows: A comparison between the source model key class members and the integrated model attribute class pool is made. The attribute class(es) most closely meeting the needs of the key class(es) for the "overlap" integrated entity class is selected. The key class(es) selected may or may not be the same as the key class(es) from the originating source model entity classes. The selection is based on the scope and viewpoint of the integrated model, and on meeting the semantic intent of the source models.

Multiple (non-synonymous) relation classes are resolved as follows: For those entity classes which have more than one relation class, the inheriting entity class is examined to determine if multiple occurrences of the inherited attribute class are required to identify the entity class. If concurrent multiple occurrences of any attribute classes are required to identify each instance of the entity class and maintain its semantic intent, then the multiple relation classes are retained. If concurrent multiple occurrences are not required, then one or more relation class and relation class label is probably not required and should be eliminated. Where a relation class or relation class label is eliminated, FEOs are constructed to document the reasons for selecting the surviving relation class and relation class label.

As the key classes are migrated, appropriate entries are made on the pencil drafts of the attribute class (Phase Three) diagrams. (Figure 5-21) When all key classes have been developed and migrated, the formal attribute class diagrams (Figure 5-22) are constructed from the pencil drafts.

During the key class migration process, refinement FEOs are constructed as appropriate (along with text where required) to document any structure changes required in the model from its Phase Two representation.

The source model views (projections) from Phase Two are revised to reflect any changes resulting from the Phase Two review cycle and the Phase Three key class assignments and key class migration.
An Attribute Class/Entity Class Index is constructed. This index lists each attribute class used in the integrated model, its owner entity class and the inheriting entity class(es) and if the attribute class is: 1) owned key class (OK); 2) inherited key class (IK); 3) owned non-key (O); or 4) inherited non-key (I). An example is shown in Figure 5-23.

As appropriate in Phase Three, kits are prepared for review and comment. A typical Phase Three review kit will consist of materials from the following list:

- Kit Cover Sheet
- Strategic Objective
- Source Model Views
- Entity Class Pool
- Source Model Entity Class Cross Reference
- Entity Class Node Cross Reference

Figure 5-22. Attribute Class Diagram: Department
<table>
<thead>
<tr>
<th>ATTRIBUTE CLASS</th>
<th>OWNER ENTITY CLASS</th>
<th>INHERITING ENTITY CLASS</th>
<th>USE CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 Account No. 1</td>
<td>54 Account</td>
<td>70 Fab lot Sub Account</td>
<td>N</td>
</tr>
<tr>
<td>16 Activity Name</td>
<td>5 Statement of Work</td>
<td>19 Activity Policy</td>
<td>N</td>
</tr>
<tr>
<td>19 Method Use</td>
<td>31 Released Engng. Drawing</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>31 Resource Plan</td>
<td>1 Section Activity</td>
<td>90 Activity Policy</td>
<td>Y</td>
</tr>
<tr>
<td>17 Change No.</td>
<td>43 Bill of Material Item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 BOM Item No.</td>
<td>5 Statement of Work</td>
<td>19 Activity Policy</td>
<td>N</td>
</tr>
<tr>
<td>7 Configuration Baseline No.</td>
<td>10 Configuration Baseline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 Confirmed Item Serial No.</td>
<td>2 Confirmed Item</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Contract No.</td>
<td>6 Contract</td>
<td>19 Activity Policy</td>
<td>N</td>
</tr>
</tbody>
</table>

Figure 5-23. Attribute Class/Entity Class Index

- Attribute Class Pool
- Source Model Attribute Class Cross Reference
- Attribute Class/Entity Class Index
- Entity Class Sets, each of which may consist of:
  - One subject Attribute Class diagram
  - Subject Entity Class definition
  - Some number of Relation Class definitions applicable to the subject Entity Class
  - Attribute Class Definitions (for owned Key Class members)
  - Refinement Alternative FEOs

5-23
A typical Phase Three kit should contain between 30 and 50 pages of material. It should require no more than one or two hours to review.

To summarize, Phase Three produces the following products.

- Attribute Class Pool
- Attribute Class Cross Reference
- Key Attribute Class Identification
- Key Attribute Class Definitions
- Attribute Class Diagrams
- Attribute Class/Entity Class Index
- Revised Source Model Views
- Refinement Alternative FEOs

5.6 Phase Four

In Phase Four, the integrated model is populated with its non-key attribute classes. Phase Four focuses attention on the attribute classes which were not utilized as members of a key class in Phase Three. The source models are used to provide guidance for the population of the non-key attribute classes.

The attribute classes not used as key classes in Phase Three are assigned to an integrated entity class based on their usage in the related source model entity class. (Figure 5-24). The assignment of non-key attribute classes to integrated entity classes may not correspond directly to the source models, because the scope and viewpoint of the integrated model may differ from the source models. The assignment of non-key attribute classes to the integrated entity classes must maintain, however, the semantic intent of the source models. The assignment in most cases will be obvious and should present no difficulty.

As each attribute class is assigned to an integrated entity class, the "no-null," and "no-repeat" rules are applied. Refinements are made as necessary in accordance with IDEF1 methodology to resolve the "no-null" and "no-repeat" violations. New integrated entity classes which emerge as a result of refinement of the "no-null" and "no-repeat" rule violations are added to the Entity Class Pool and source model entity class cross reference, attribute class diagrams are constructed, and source model projections updated as required.
When population of the integrated entity classes with non-key attribute classes is completed, the Attribute Class/Entity Class Index is updated to document ownership of the non-key attributes.

As appropriate in Phase Four, kits are prepared for review and comment. The contents of a Phase Four kit will be essentially identical to the Phase Three review kits, but the content will reflect the distribution of non-key attribute classes, and any structural changes resulting from Phase Four refinement. Phase Four kits consist of materials from the following list:

- Kit Cover Sheet
- Strategic Objective
- Revised Source Model Views
- Revised Entity Class Pool
- Revised Source Model Entity Class Cross Reference
- Revised Entity Class Node Cross Reference
Revised Attribute Class/Entity Class Index

Revised Entity Class Sets, each of which may consist of:

- One refined subject attribute class diagram
- Subject entity class definition
- Some number of relation class definitions applicable to the subject entity class
- Refinement alternative FEOs

Revised/Refined Attribute Class Definitions (Key and Non-Key Attribute Classes)

A typical Phase Four kit should contain between 30 and 50 pages of material. It should require no more than one or two hours for review.

In summary, Phase Four, rather than producing an appreciable quantity of new material, concentrates on the further delineation of already established materials.

5.7 Conclusion

With the completion of Phase Four of this process, an integrated information model will have been produced. If the methodology has been adhered to throughout the model's development, the integrated model will represent a stable, integrated, information structure from the source models from which it originated can be "projected" in their revised form. This new "integrated" model provides a stable foundation for the future integration of additional information into the enterprise information structure.
APPENDIX A

SUMMARY SYSTEM 0
PUBLICATION

1. INTERPRET FABRICATION STEP
   - Finished part reqmts
   - Assignments
   - Work to be done by step

2. SELECT SPECIFIC MACHINE TYPE
   - Equipment types
   - Machine type selection
   - Part & tool setup

3. DETERMINE SETUP
   - Part initial form
   - Equipment description
   - Info on tools available & possible

4. SPECIFY TOOLS
   - Tool ident, tool specs

5. DEVELOP DETAILED INSTRUCTIONS
   - Detailed fab step instructions
   - Detailing problems
Note: All of 62 is included except A623 (263) which have a limited impact on the outputs of A623.
PUBLICATION

LOCATE & PULL MATERIAL

C1 pull list

C2 work packages

open stock material

production kit orders

spares/repair & retrofit kits

Now SHORTAGE

disbursement, shortage records

shortage reports

06 04

04 01

inventory

delivers against shortages

organizes materials

kits, needed items

03 items & tools for kits

parts & tools for kits & spares

prepare & ship spares and kits

02 spare parts, repair and retrofit kits