The Practical Guide to Business Process Reengineering Using IDEF0

Clarence G. Feldmann

Foreword by John V. Tieso

DH

The Practical Guide to Business Process Reengineering Using IDEF0

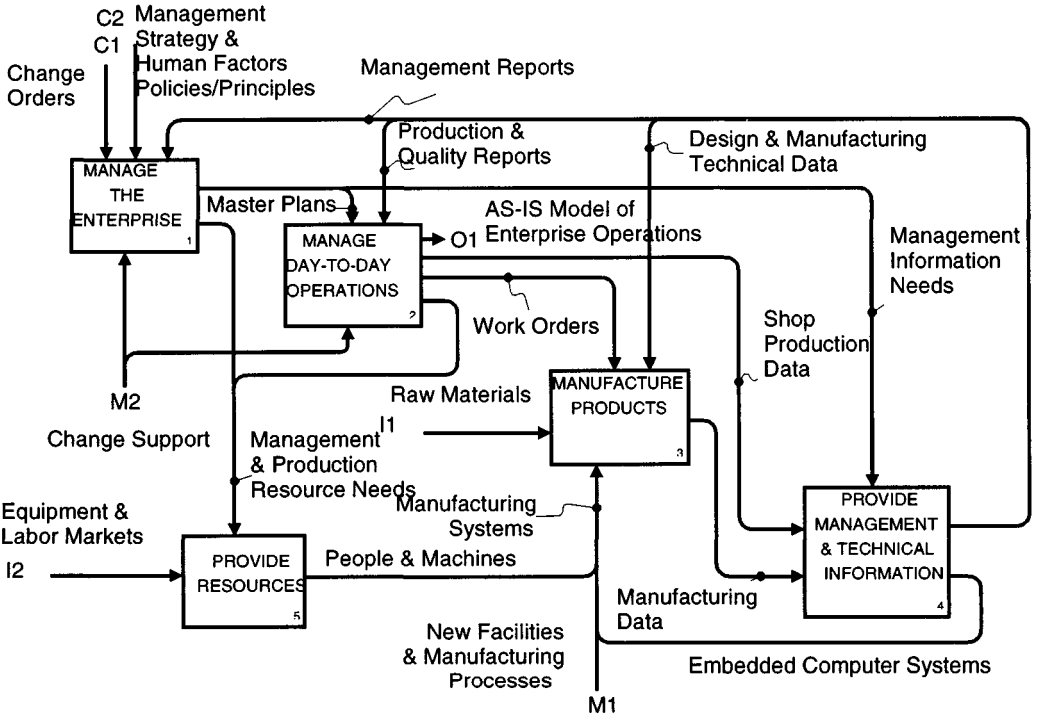

Figure 3-8: A Typical IDEF0 Diagram.

IDEF0 graphics are undeniably straightforward and that quality is, in my view, one key to their success. Such elaborations as additional box shapes, dotted arrows, and formalisms for arrow branching structures have been experimented with by modelers since the inception of IDEF0. It seems only natural to conclude that such elaborations would provide additional expressive capabilities to the analyst.

However, I have learned to resist the temptation to elaborate the syntax, since this makes the graphics too complex to fulfill the essential role of the model to communicate. At the early stages of project planning, communication is the key element, and the simple graphics that are quickly learned and employed must remain a basic element of IDEF0.

Levels of Abstraction in IDEF0 Models

As stated, the breakdown approach used in IDEF0 modeling decomposes selected activities into finer and finer levels of detail. The resultant model set includes sufficient detail to plan and control the implementation of changes to the enterprise.

An IDEF0 model may also contain multiple levels of abstraction. To understand what is meant by "level of abstraction," let's consider a typical enterprise. Each level of management has its natural level of abstraction. Top management must think in broad terms, taking into account many viewpoints as well as abstractions of the details of what is going on in the enterprise. An upper-level manager cannot possibly understand every detail of all of the activities in the entire enterprise, but he must abstract business processes and characteristics that are important at his level of abstraction. He must consider such things as overall profit, hiring policies, or the quality of the enterprise's product line, rather than the specific factors that go into these high-level abstractions.

Conversely, a manager at a lower level needs fine-level detail about a specific topic from a single viewpoint, and he may get uneasy thinking in general terms. His level of abstraction is concerned with such things as the cost of raw materials, the capabilities of specific staff members, or the quality of the service for which he is responsible.

The model's level of abstraction is independent of its level of detail. That is, after selecting a level of abstraction to model, the author may decompose the activities to whatever level of detail he requires to satisfy the purpose of the model. However, if the purpose of the model requires information at a different level of abstraction, no amount of further detailing at the present level of abstraction will provide the needed information. Modeling at all required levels of abstraction must be completed to satisfy the enterprise analysis requirements.

The IDEF0 concept of a mechanism is different from the concept of level of abstraction. In IDEF0, a mechanism represents who performs the activity and what tools (software packages and equipment) are required to perform the activity. In other words, the mechanism identifies the resources needed to perform the function. A mechanism is therefore at a lower level of abstraction than the activity box it implements.

The fact that the mechanism depicts a lower level of abstraction does not mean that it cannot be modeled in IDEF0. It just means that the mechanism is shown as an arrow entering the bottom of the box, and that a separate model must be examined to understand how the mechanism works (see Figure 3-9).

The IDEF0 diagram in Figure 3-9 shows Activity A23 (Edit Document) using the word processor support mechanism. The IDEF0 model for the word processor is shown at the lower-right corner of Figure 3-9. What this tells us is that any word

processor mechanism may be modeled and plugged into the diagram to show the precise activity performed when editing a document with any variety of word processors. This shows *how* the editing is done using the mechanism, and therefore indicates that there is a drop in level of mechanization from *what* is done to the document by the EDIT activity.

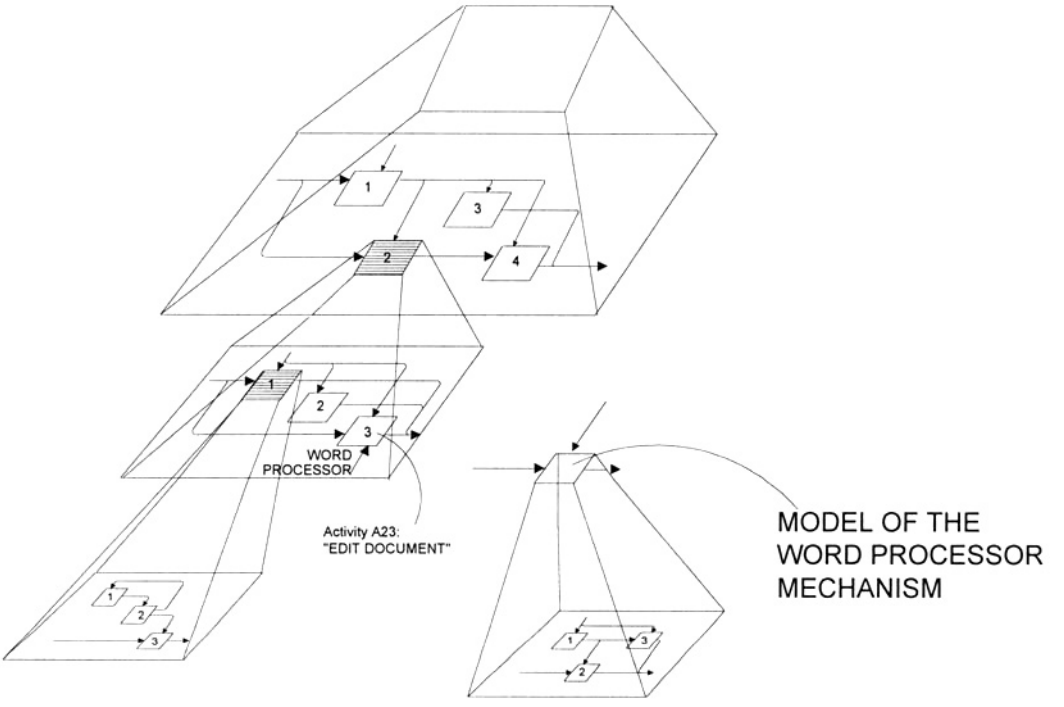

Figure 3-9: Mechanism at a Lower Level of Abstraction.

To accomplish the reengineering of an enterprise, an analyst may need to show multiple levels of abstraction and several levels of mechanism, as well as how those are integrated to perform the processes of the enterprise. Showing all this provides a clear, accurate *big picture* of the overall operation of the enterprise.

In addition to an IDEF0 process-oriented analysis, various additional analysis methods may be applied, using the IDEF0 model as a baseline. For example, the analyst may need to know where costs and labor are being expended, in order to estimate the return on investment of potential changes or to understand the impact of introducing new technology. Various analyses, such as activity-based costing (ABC) and work flow simulation, are typically applied using the IDEF0 model as a baseline.

The Role of Data Analysis Compared to IDEF0 Function Analysis

Detail needed regarding the processes requires not only IDEF0 process models, but also precise information about the arrow content. This requires a careful analysis of data.

An analyst may be confronted with a tidal wave of raw data when he first gathers information about an enterprise. He typically starts by finding out the precise meaning of the terms that are key to the system's operation; then he documents these definitions. Once he has done that, he has a solid basis for initiating discussion with the staff about the detailed operation of the system. But even before analyzing the system in detail, he must consider the system functions and their interactions to understand the need, the purpose, and the objective of the relevant data.

Not all analysts take this view. In fact, some prominent leaders in the database community have stated that there is no need to look at the functions and activities at all. One database proponent wrote that the development of a function model actually gets in the way of the programmer and should be avoided if possible, further proclaiming his First Rule of CASE: "Only one diagramming method is needed—the data model—and code can be generated automatically from a data model."[1]

This statement is true if the automated system to be built performs simple, well-known processes such as basic banking transactions (for example, put money in checking account, withdraw money from checking account, transfer money to savings account, and so on). These activities are so familiar that an IDEF0 model would not be helpful, and design of the data structure of the system can be started almost immediately. The statement is not true, however, if the software to be built has a complex processing aspect, such as that for an avionics system or a computer's operating system.

The field of automated programming would also lead one to believe that analysis, design, and programming are outmoded—as if the system will do all this for you and produce a running computer program. However, the type of system that is implied here is one that simply queries a database and displays answers on a screen. Automatic programming has a long way to go before it can build an avionics system, a computer operating system, or a compiler. And haven't we slipped into the solution domain? Automated programming tools are of no help if you are trying to reengineer a business operation and have not yet understood what the system will look like.

---

[1] Clive Finkelstein, *An Introduction to Information Engineering: From Strategic Planning to Information Systems* (Reading, Mass.: Addison-Wesley, 1989), p. 376.

Regardless of the system to be analyzed and the comparative importance of the data versus the activity analysis, at some point the data must be analyzed. To perform this portion of the analysis, the analyst must understand several important characteristics of the data.

Although there are many data characteristics that could be analyzed, two are critical: data dynamics and data structure. Data dynamics can be depicted by SADT data models, using a portion of SADT that was not adopted by the Air Force for IDEF0. Data structure may be modeled using IDEF1X, where the term "structure" signifies all non-dynamic data information, such as its attributes and its relationship to other data elements.

A type of function model, the SADT data model shows the breakdown of the kinds of data in the system, with arrows indicating which activities produce which data elements and which activities use the data for what purpose. This gives a dynamic picture of the data being created, used, and merged with other data to form new data entities. The SADT data model is a natural companion for an IDEF0 or SADT activity model, and it is useful for analyzing how the system manipulates data and what is affected when changes are implemented. It also provides a validation of the activity model. In fact, most data models developed using SADT result in corrections to the activity model!

The IDEF1X model provides all of the structural information about the data. Ultimately, relating the need for and the usage of specific attributes of data by the activity boxes of an IDEF0 model is the key to integrating the two models.

CHAPTER 4

IDEF0 Graphic Language Syntax and Semantics

The syntax and semantics for both IDEF0 diagrams and IDEF0 models are precisely defined in the FIPS for IDEF0.[1] The summary presented in this chapter is intended to provide sufficient practical detail for the IDEF0 modeler, but modelers should refer to the FIPS itself for a more complete definition.

As we have seen in previous chapters, IDEF0 models are made up of diagrams arranged in a hierarchical structure. IDEF0 diagrams are made up of boxes and arrows, where the boxes represent the happenings (activities) and the arrows represent the interfaces between those happenings—the things. Each diagram tells a story about a portion of the system being modeled. The top-most diagram represents the entire system, and each box on this top diagram is decomposed into subordinate models at the next finer level of detail. This succession of decompositions forms the hierarchical model structure.

Each IDEF0 diagram is presented on a sheet of paper called a diagram form. In the following section of this chapter, the elements of an individual IDEF0 diagram (the box and the arrow) are defined. In succeeding sections, these elements are used to create diagrams, and the diagrams are then structured to form a complete IDEF0 model.

---

[1] *FIPS PUB 183: Integration Definition for Function Modeling (IDEF0)*, U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology (Washington, D.C.: 1993).