
2.0. BUILDING MANAGEMENT TOOLS

2.1. APPROACHES FOR BUILDING TOOLS

2.1.7. LOGICAL DATA ANALYSIS

2.1.7.1. DATA, INFORMATION, AND IMAGE STORAGE—LILLA CABOT-PERRY

**2.1.7.2. IMAGES AND DATA ARE PART OF INFORMATION RESOURCE
MANAGEMENT**

2.0. BUILDING MANAGEMENT TOOLS

2.1. APPROACHES FOR BUILDING TOOLS

2.1.7. LOGICAL DATA ANALYSIS

2.1.7.3. NORMALIZING DATA BASES

2.1.7.3.1. ANALYZING DATA

To evaluate the roles data play in the organization, we analyze the sources and sinks of data, the data stores, and the information flows.

In building any management tool, we ultimately must deal with data. Data is the stuff we make information from. We use information to support decision making. We've approached data from the starting point—from decisions. We used the management system model. Considering data and information, we'll step through the majority of the following steps.

1. Domain
2. Context
3. Analyze relationships (partitioning technique)
4. Information flows (DFD and data dictionary)
5. Output formats
6. Input methods
7. Information sources (and sinks)
8. Stores (files)
9. External entities
10. Conversion processes
11. Synthesize relationships (layering technique)
12. Design flows
13. Design sources
14. Design conversion processes
15. Test relationships

Notice several things from this list of steps. First, the steps are information and data flow oriented. That is, they don't include decisions. We must figure out what the decisions are we'll use the information for. And we do that by doing charts like logic charts and work flow diagrams. The decisions feed into the steps I've just listed at the point where we start dealing with analyzing the relationships internal to the domain of responsibility. Second, the analysis steps tend to carry us from a physical perspective to a logical one. Third,

the synthesis steps tend to carry us from a logical perspective back to a physical one. That is, when we start, we use a physical perspective in learning what we need from the user. Then, when we're ready to turn everything back to the user, we must go back to a physical perspective again.

Looking at the 15 steps and thinking about what we've talked about so far, we're ready to start thinking about sources (and sinks) of data. Data are obviously generated either internal or external to the domain of responsibility. And they disappear either internal or external to it. Often, as we deal with data, we want to store them and save them for a while—maybe for a short while, maybe for a long while. So we can get data either from a source or from a store. And we can send them to either one. Figure 2.1.7.3.1. shows the sources and sinks of data as we make information.

In logical data analysis, we can look at three concepts:

1. Document analysis
2. External source/sink analysis
3. Data store analysis

In document analysis, you study the documents (or portrayal formats) in the domain of responsibility. They have data displayed on them. How good are the data? Are there redundant data? We can also ask questions like, "Are the data in the right place?" In logical data analysis, we're interested in the substance and redundancy of data, not the form of data.

In analyzing external sources and sinks, you study the veracity, timing, and use of data,

based on where you get the data from or who you send them to. In all of this, we're interested in "data about data." Consider a piece of data indicating a milestone has slipped two months. Who got that datum (data about the data)? When did that datum get put in the system?

In data store analysis, you study the simplicity and redundancy of putting data into a place you can faithfully retrieve them from. Most of data analysis will focus on this analysis, but we can't overlook the others. We like to deal with data store analysis because it's easiest to define and describe.

These analyses can be carried out in great detail. The amount of detail depends on your objective when doing the analysis. A cursory or simple analysis will help find big overlaps or gaps in the data and will help you understand the management process, the domain of responsibility, and how the flow of information and the sources and sinks of data work. A detailed or complex analysis will help you design a more efficient and easy-to-use information system. Much like the management element (a concept I'll describe shortly), good analysis here tends to help us converge when things get complicated.

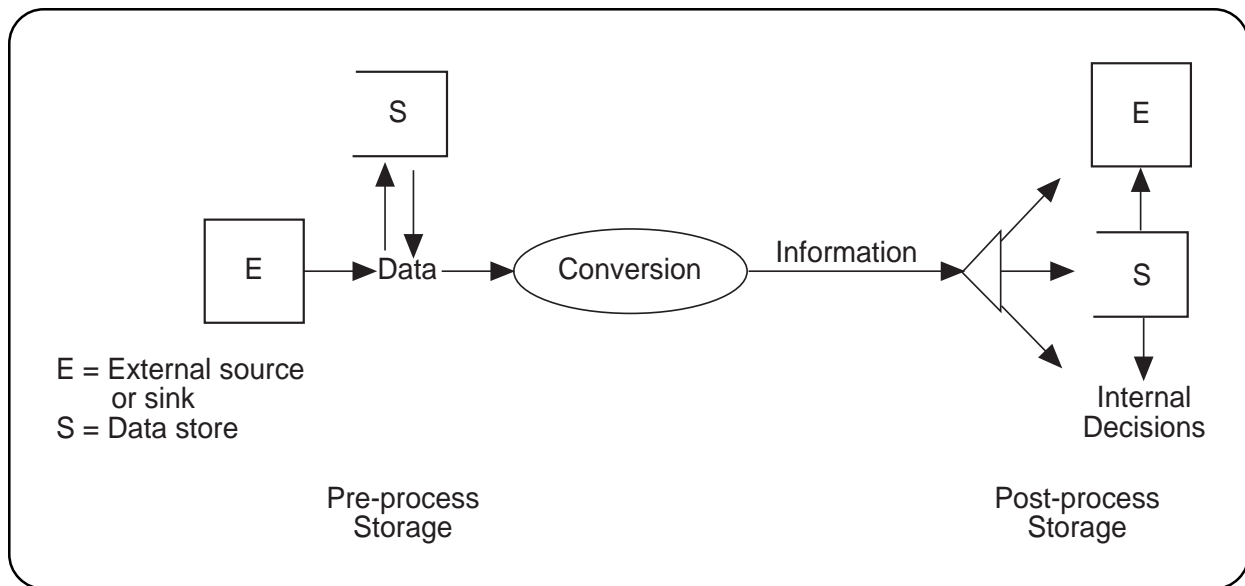


Figure 2.1.7.3.1. The sources and sinks of data and the data stores fit into the data to information conversion process.

2.1.7.3.2. DOCUMENT ANALYSIS

One quick way to improve data integrity and reduce steps in an organization is to study and streamline the many types of documents the organization has accumulated over the years.

In many organizations you'll find a large number of documents. Many of the data will be repeated from document to document. Some of the data will never be used. For most of the data, you'll know nothing about who put them there, or when, or how old the data are.

Most output formats in a domain of responsibility evolve. Also, the information system evolves as different output formats are needed.

Consider this example. While you're in college, you decide to sell tee-shirts on consignment. You start with an order form. That's all you need to keep track of what you're doing. So you make two data stores. One keeps track of the shirts you've sold (sizes, colors, etc.). The other keeps track of your customers (name, address, maybe color and size, etc.). From these data stores you can produce documents listing which shirts make up most of your business or listing your best customers so you can send them a Christmas card.

Then your business grows. You need some additional students (sales people) to keep up with all the business. You also have to keep some shirts on hand (the fast moving sizes, styles, and colors). Now you need data stores for sales people and inventories. You'll need merchandising and accounts payables documents. Soon you'll need to have payroll, accounts receivables (no longer cash on the barrel head), withholding taxes, and other documents. As you do business, you'll want to add to various data stores so when you need to

aggregate data, you can. You'll have more data stores and more documents.

Someday, you'll have other personnel in addition to salespeople—secretary and stock clerks, for example. So you have different types of wages. Some get fringe benefits (insurance, paid vacation, etc.). How much vacation is accumulated? As you get bigger, some of your people will get perks (e.g., a company car).

You didn't analyze all this. It just happened. Your system evolved. You developed your information portrayal formats as you needed them. Many of your formats contain the same data. In some cases you get data for several documents from the same data store. In most cases, you don't. You need to do a document analysis to see what information you still need. In some cases, you don't need to know shirt color like you did when you operated your business out of the trunk of your car. When you developed the merchandising system, the shirt color was incorporated in the item number for the shirt. You're keeping redundant data—sometimes on the same document—or at least on several documents.

These redundant data beget problems with data integrity. Each time a datum changes, you have to find every place it occurs. Then you have to change that datum. Chances are you'll miss one or more occurrences of a changed datum. Then you won't know which value for the datum is the right one.

2.1.7.3.3. DATA STORE ANALYSIS

If we don't eliminate redundancy and complexity in data stores, we'll surely reap data integrity problems.

In the shirt business example in Module 2.1.7.3.2., we talked about building separate tools for small functions in the bottom-up approach. We built one information system for one separate function. The information system had its own database. Then we did another information system for another function with another database. We used the classical bottom-up approach (see Module 2.1.4.) for developing our information system. With two databases we had update problems and data integrity problems—typical problems in the bottom-up approach. When we went to more systems for more functions, the problem increased.

Now, we'll look at a top-down approach where we do several functions simultaneously. We'll find several files in our one database (perhaps one, maybe a couple). The database will be kept in one mechanism—not a file cabinet and rolodex, but a computer database. Let's see what redundancy plagues us even now. We want to remove redundancy from any files, whether in a file cabinet, rolodex, and/or on a computer.

At Management Systems Laboratories, no one trusts the central filing cabinets—mostly because we don't lock them. Locked filing cabinets make people angry. Unlocked file cabinets mean we don't know what has been put in, removed, or replaced. We don't know how to share data. So everyone has their own filing cabinets. Things are filed in several places. Sometimes, a document is updated in one of the places. The others stay the same. Then, the document you get depends entirely on which filing cabinets you go to—not on

which document is most current or accurate.

By studying the idea of normalizing data files (data stores), we'll learn a technique for eliminating redundancy and promoting integrity. The technique also makes the files simpler (always a valuable thing to do). We'll apply normalization to data stores, but we can extrapolate the ideas to document analysis and other issues where we want to eliminate redundancy.

Redundancy is bad in a computer because in the computer data are out of sight and we forget we may have some data (or derivatives of those data—e.g., hours, pay rate, and pay) in several places. Since we can't readily see the situation, we forget it.

We've studied information flows and information conversion processes. Now we'll study data stores.

We can apply two main criteria to data stores so we can see if their components and organization are best for the information system we're interested in. One of these criteria is simplicity. We want both the organization of and the access to a data store to be as simple as possible. Usually that means we'll implement the data store as a simple sequential or direct-access file. That is, the elements in the data store are referenced by a primary key and we have no repeating groups of data in the data store. A primary key consists of one or more data elements that identify uniquely the occurrence of a data structure in a data store. Your social security number might be the key for a data store containing data about you. A data

store is simpler if it doesn't have alternate keys or pointers to link records, and if it doesn't require implementation as variable-length records. The criterion for logical data organization is always simplicity over complexity.

The user thinks in terms of information files and records. The data processor thinks in terms of data, groups, records, and blocks. Before we discuss data stores in detail, we need some definitions to relate these terms. I've shown the definitions and the hierarchical nature of the terms in Figure 2.1.7.3.3.

The second criterion for logical data design is non-redundancy. We have redundancy, for the most part, when the same data element exists within two or more data stores. Redundancy can threaten the integrity of a system. If you change the value of a data component in one file, you must change the same element in all other files. We also have redundancy when the same data appear in different forms within the same data store—when two or more data components within a data store give the same information.

DATABASE - aggregate of files to meet information system requirements

FILE - related records or blocks

BLOCK - two or more records retained in a particular storage medium such as a file cabinet or computer tape

RECORD - a collection of data elements related to a common identifier such as a person, machine, place, or operation

GROUP - two or more data elements that are logically related and must appear to form a complete unit of meaning (street number and name, first and last name of a person)

DATA ELEMENT - the lowest level of the data structure and the only one with which a specific value may be associated (age, part number, etc.)

Figure 2.1.7.3.3. *The convention for showing the aggregations of data into storage units helps us dig into a data store to find the datum we want.*

2.1.7.3.4. NORMALIZING DATA STORES.

An example will help us track down the potential for data integrity problems and build a data store that has no redundancy.

In the Management Systems Laboratories' personnel accounting system you saw when we discussed information flow diagrams, or data flow diagrams DFD's, in Module 2.1.6.6., we focused on information flows, information conversion processes, and external entities. We looked at that example from a physical perspective. Now let's look at a companion example—the system MSL uses to assign people to tasks and account for the time they spend on those tasks. We'll look at this example from a more logical perspective and we'll see more detail in flows and processes. And we'll see some data stores. I've shown the DFD for the personnel task assignment and execution system in Figure 2.1.7.3.4.a.

The domain of responsibility for this system is also in MSL's operations group. In Figure 2.1.7.3.4.a., you don't see much about people and places. This DFD is logical. You see three external entities to the domain. The group managers are responsible for their people. They hire and fire them, give them raises, and find contracts (or projects) for them to work on. The group managers negotiate with the contract managers to find the best place for their people to make a contribution to MSL's contracts. The contract managers are responsible to meet the requirements of certain contracts. MSL's people are one of the resources the contract managers need to meet those requirements. They divide their contracts into tasks and look for people to do the work of the task. The employees work for the group managers and are assigned to contracts under the direction of the contract managers. Usually the contracts they're assigned to are in their group.

Eight data stores are shown in Figure 2.1.7.3.4.a. The figure was produced by talking to (interviewing) the people involved. (We used our information gathering skill so we can get to the information flow diagramming skill and ultimately get to the skill for eliminating redundancy in data stores.) After doing this first-draft figure, I haven't yet reviewed it with those people and verified its accuracy. But the figure is good enough to analyze data stores. Knowing how this system has evolved over the years at MSL, I expect there's lots of redundancy in the files. Also the files aren't very simple. Usually, in a DFD, we don't label the flows out of and into a data store. I've labeled some of these flows in Figure 2.1.7.3.4.a. to help describe what's in the files. In this case, I've put those labels in parentheses. The flows between information conversion processes and external entities in Figure 2.1.7.3.4.a. are labeled as usual.

Generally speaking, what happens physically in this system is that the contract manager gets a new contract and defines tasks he or she wants people to work on. The group manager has people that need to be put to work. The manager of this domain identifies all tasks and puts them in a file. He or she also identifies all available people and puts data about them in a file. Then, on the advice of the group and contract managers, the employees are assigned to tasks and reports are sent to the group managers about what their people are working on and to the contract managers about who's working on their tasks. The employees get a report showing what tasks they can work on and charge their time to. As the employees work on their tasks, they fill in time sheets,

which they submit to the manager of this domain each week. The assigning of people is the formulation side of the exercise. The charging to tasks is the execution side of the exercise. When employees are charged to tasks, the contract managers need to know exactly who and how many hours were charged to their tasks. The employees and their group managers need to know exactly what tasks the people were charged to and for how many hours.

Figure 2.1.7.3.4.b. shows a partial data dictionary for the data stores in Figure 2.1.7.3.4.a. Each data store is an iteration of a data structure. Thus, the normalization of data stores is essentially equivalent to the normalization of a set of data structures. You can glance at Figure 2.1.7.3.4.b. and see a lot of redundancy among the data stores. Look for task name for example.

We design data stores to support the information conversion processes. We need to derive the best logical design for this set of data stores. In this analysis, we'll replace the group of existing data stores by its logical equivalent. The result is a set of simple data stores containing no redundant elements.

The procedure used to derive this logical structure is called normalization. In general, normalization produces the simplest, most straightforward organization of data elements into component data stores. Normalization should produce a set of data stores containing non-redundant data elements accessible through use of unique primary keys. The keys are highlighted by showing them in bold type. I'll illustrate the procedure for MSL's personnel assignment and execution system.

STEP 1: PARTITION EACH DATA STRUCTURE THAT CONTAINS REPEATING GROUPS OF DATA ELEMENTS.

Take each repeating group of data elements

and form two or more data structures without repeating groups. The newly formed data structures must accomplish the same purpose as the original repeating group of data elements. After we complete this step, the original data structures of Figure 2.1.7.3.4.b. will be converted into a state known as the first normal form. From Figure 2.1.7.3.4.c. you can see the result of step 1. For each data structure that contained a repeating group, the repeating group was removed and set up as a separate data structure. The key for the new structure was formed by concatenating the key for the original data store with the key for the repeating group. The key for the original file was retained as the key for the data structure without its repeating group. Again, in the figure the keys are highlighted by using bold type.

Figure 2.1.7.3.4.d. illustrates the original CONTRACT LOADING FILE. In Figure 2.1.7.3.4.d., I've displayed hypothetical data the way the data dictionary for the CONTRACT LOADING FILE indicates they should look. The first normal form yields three new data structures that accomplish the same purpose as the original. The new data structures are shown in Figure 2.1.7.3.4.e. They are the LOADED CONTRACT FILE, the CONTRACT TASKS FILE, and the CONTRACT EMPLOYEE FILE. I've tried to select meaningful names for the new files. The names are different from the original and from any other file. From Figure 2.1.7.3.4.e., you can see the new files are simple and are linked together through their keys.

STEP 2: VERIFY THAT EACH NONKEY DATA ELEMENT IN A STRUCTURE IS FULLY FUNCTIONALLY DEPENDENT ON THE PRIMARY KEY.

This step places the set of data structures in the second normal form. In doing this step, we deal only with those structures that are identified by concatenated keys. We accomplish the

work of this step by verifying that each nonkey data element in a data structure is dependent on the full concatenated key, not just on a partial key. That is, each element should require the entire key as a unique identification. Instead, if a data element is determined uniquely by only a part of the key, the element should be removed from the structure and placed in a structure of its own. In some cases, one of the resulting structures contains only key information. This is both permissible and necessary.

From Figure 2.1.7.3.4.f. you can see the result of step 2. Figure 2.1.7.3.4.g. shows the CONTRACT TASKS FILE in the first normal form and then shows the results of changing to the second normal form. In the second normal form we've gotten the TASK 1 FILE and the CONTRACT POSITIONS FILE.

STEP 3: VERIFY THAT ALL NONKEY DATA ELEMENTS IN A DATA STRUCTURE ARE MUTUALLY INDEPENDENT OF ONE ANOTHER.

After we've converted the data structures to the second normal form, each structure is checked to verify that each non-key data element is independent of every other non-key element in the relation. We remove duplicate data elements or elements that can be derived from other elements to place the relation in the third normal form. The only redundant non-key data element in a data structure is "task duration" in the TASK DATA FILE. To get the third normal form, we must eliminate "task duration."

STEP 4: ELIMINATE REDUNDANT DATA ELEMENTS AMONG THE DATA STRUCTURES.

After we put the set of data structures in the third normal form, there are likely to be redundancies among the normalized structure. We simply remove redundant elements. In doing this, we have to use some judgment. In making

the decision on which structures we should assemble into a composite structure, we are guided by a sense of what the object of a structure is and by what the attributes of that structure are. The object of a data structure is the entity to which the structure pertains. The attributes of an object are items of data characterizing that object. The data structure contains the attributes that pertain to one and only one object. No superfluous data elements appear in the relations, and no elements appear as attributes in any other relations. Figure 2.1.7.3.4.h. shows the results of steps 3 and 4. Three data structures are in the third normal form.

As a mechanical step, the revisions we've made to the set of data stores in Figure 2.1.7.3.4.a. must be reflected in the data flow diagram and data dictionary. The data flow diagram in Figure 2.1.7.3.4.i. is a redrawing of the original diagram from Figure 2.1.7.3.4.a. I've removed one of the information conversion processes because it dealt with creating a file we no longer need.

Advantages of the Third Normal Form

The following list summarizes the advantages of having data in the third normal form.

1. *Ease of Understanding.* In the third normal form, we present data structures in a way that operational and management users can easily understand them. The data structures are presented in simple, two-dimensional tables that do not require technical understanding on the part of the users and owners of the data.
2. *Ease of Use.* We can partition data structures further to represent any number of different logical viewpoints. Different structures can have different file organizations to allow efficient access for primary application, yet be accessible for many other secondary applications. Attributes from different files can be related with

little complexity.

3. *Ease of Implementation.* We can implement structures in the third normal form as simple files set up for either serial or direct access. Also, we can implement the structures within database systems.

4. *Ease of Maintenance.* Non-redundancies in the files reduce problems while keeping all files up to date. If we must add to, change, or delete an attribute from a file, we're sure that no other file will require the same maintenance. All nonkey attributes appear only one at a time in one place within the file structure.

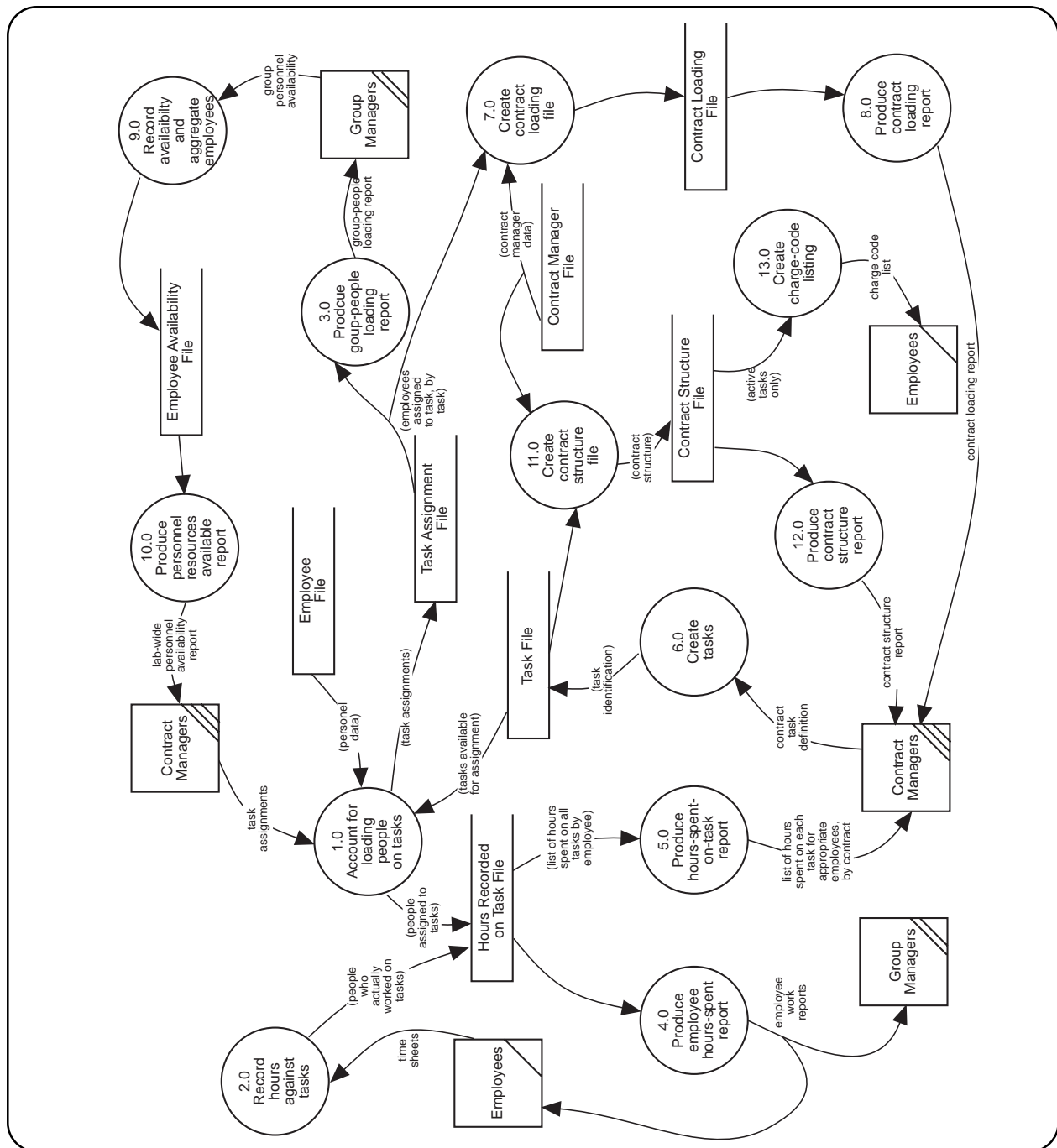


Figure 2.1.7.3.4.a. Data flow diagram for MSL's personnel assignment system.

EMPLOYEE FILE EMPLOYEE	=	{EMPLOYEE} All employees where	
	=	Employee-Number + Employee-Name + Employee-Address + Employee-Start-Date + Employee-Group-Assignment + Employee-Qualifications +	
		{ Position-Title + Pay-Rate + Fringe-Rate + Pay-Start-Date	All pay rates for employee
TASK FILE TASK	=	{TASK} All tasks available on a contract where	
	=	Task-Number + Task-Name + Task-Leader + Contract-Manager + Contract-Number + Task-Budget + Task-Start-Date + Task-End-Date + Task-Skill-Requirements	
TASK ASSIGNMENT FILE TASK ASSIGNMENT	=	{TASK ASSIGNMENT} All tasks where	
	=	Task-Number + Task-Name + Task-Leader + Contract-Manager + Contract-Number + Task-Budget + Task-Duration + Task-Start-Date + Task-End-Date + Number-of-Task-Position +	
		{ Employee-Number + Employee-Name + Employee-Qualifications + Employee-Group-Assignment	All employees on task
CONTRACT LOADING FILE CONTRACT LOADING	=	{CONTRACT LOADING} All tasks where	
	=	Contract-Number + Contract-Manager-Name + Contract-Manager-Group + Number-of-Tasks-on-Contract +	
		{ Task-Number + Task-Name + Number-of-Task-Positions +	All tasks on contract
		{ Employee-Number + Employee-Name	All employees on task

Figure 2.1.7.3.4.b. Data Dictionary for Data Stores in MSL's Personnel Assignment System.

CONTRACT MANAGER FILE	=	{CONTRACT MANAGER} All contracts where	
CONTRACT MANAGER	=	Contract-Manager-Employee-# + Contract-Manager-Name + Contract-Manager-Group + Number-of-Contracts +	
		{ Contract-Number + Contract-Name }	All contracts assigned to contract manager
CONTRACT STRUCTURE FILE	=	{CONTRACT} All contracts where	
CONTRACT	=	Contract-Number + Contract-Name + Contract-Manager-Name + Contract-Manager-Group + Number-of-Tasks-on-Contract +	
		{ Task-Number + Task-Name + Task-Status }	All tasks on contract
EMPLOYEE AVAILABILITY FILE	=	{AVAILABILITY} All employees where	
AVAILABILITY	=	Employee-Number + Employee-Name + Employee-Qualifications + Percent-Time-Available	
HOURS RECORDED ON TASK FILE	=	{HOURS RECORDED} All tasks for all employees where	
HOURS RECORDED	=	Task-Number + Task-Name + Task-Duration-in-Weeks	
		{ Employee-Number + Employee-Name + Week-Number + Week-Name + Hours-Spent + Validity-of-Employee-on-Task }	All employees on the task

Figure 2.1.7.3.4.b. (cont.) Data Dictionary for Data Stores in MSL's Personnel Assignment System.

DATA STRUCTURES IN FIRST NORMAL FORM

ORIGINAL DATA STRUCTURES

DATA STRUCTURES IN FIRST NORMAL FORM

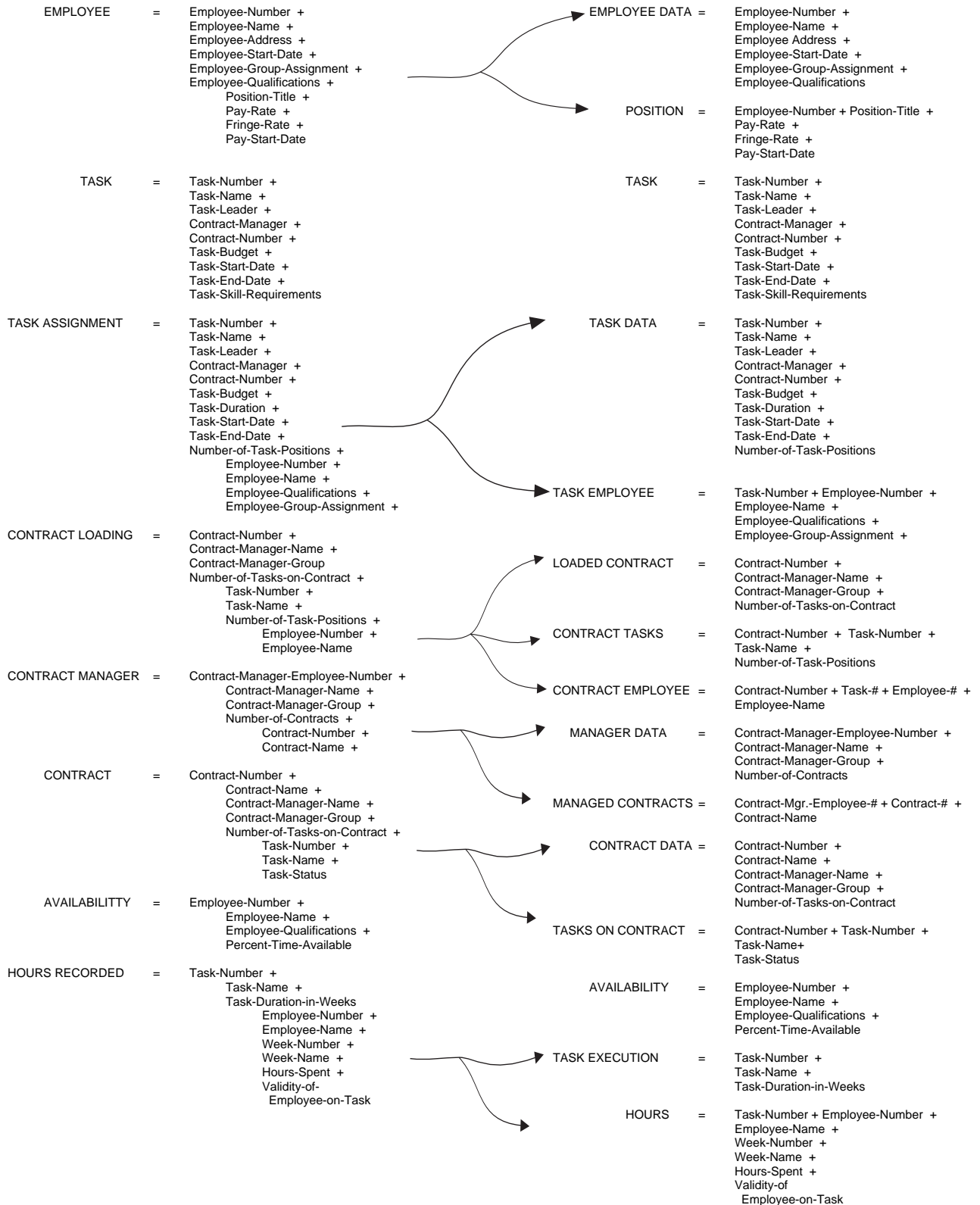


Figure 2.1.7.3.4.c. MSL's Personnel Assignment System Data Structures in First Normal Form.

CONTRACT LOADING (Contract-Number + Contract-Manager-Name + Contract-Manager-Group + Number-of-Tasks-on-Contract + {Task-Number + Task-Name + Number-of-Task-Positions + {Employee-Number + Employee-Name}})

9-62	John Doe	Systems Group 3	9-62-01	Do Analysis	3	0138	Jim Young
						2156	Linda Steele
						1521	Susan Blank
			9-62-02	Evaluate Results	2	1356	George Smith
						0511	Betty Jones
			9-62-03	Write Report	1	0138	Jim Young
9-75	Peter Brown	Materials Group 2	9-75-01	Write Code	2	2156	Linda Steele
						3571	Henry Jordan
			9-75-02	Test Code	1	3571	Henry Jordan
9-81	John Doe	Systems Group 1	9-81-01	Write Report	1	0852	John Doe

Figure 2.1.7.3.d. *Original CONTRACT LOADING FILE.*

LOADED CONTRACT (Contract Number + Contract Manager Name + Contract Manager Group + Number of Tasks on Contract)

9-62	John Doe	Systems Group	3
9-75	Peter Brown	Materials Group	2
9-81	John Doe	Systems Group	1

CONTRACT TASKS (Contract Number + Task Number + Task Name + Number of Task Positions)

9-62	9-62-01	Do Analysis	3
9-62	9-62-02	Evaluate Results	2
9-62	9-62-03	Write Report	1
9-75	9-75-01	Write Code	2
9-75	9-75-02	Test Code	1
9-81	9-81-01	Write Report	1

CONTRACT EMPLOYEE (Contract Number + Task Number + Employee Number + Employee Name)

9-62	9-62-01	0138	Jim Young
9-62	9-62-01	2156	Linda Steele
9-62	9-62-01	1521	Susan Blank
9-62	9-62-02	1356	George Smith
9-62	9-62-02	0511	Betty Jones
9-62	9-62-03	0138	Jim Young
9-75	9-75-01	2156	Linda Steele
9-75	9-75-01	3571	Henry Jordan
9-75	9-75-02	3571	Henry Jordan
9-81	9-81-01	0853	John Doe

Figure 2.1.7.3.4.e. *Components of MSL's personnel assignment system in the first normal form.*

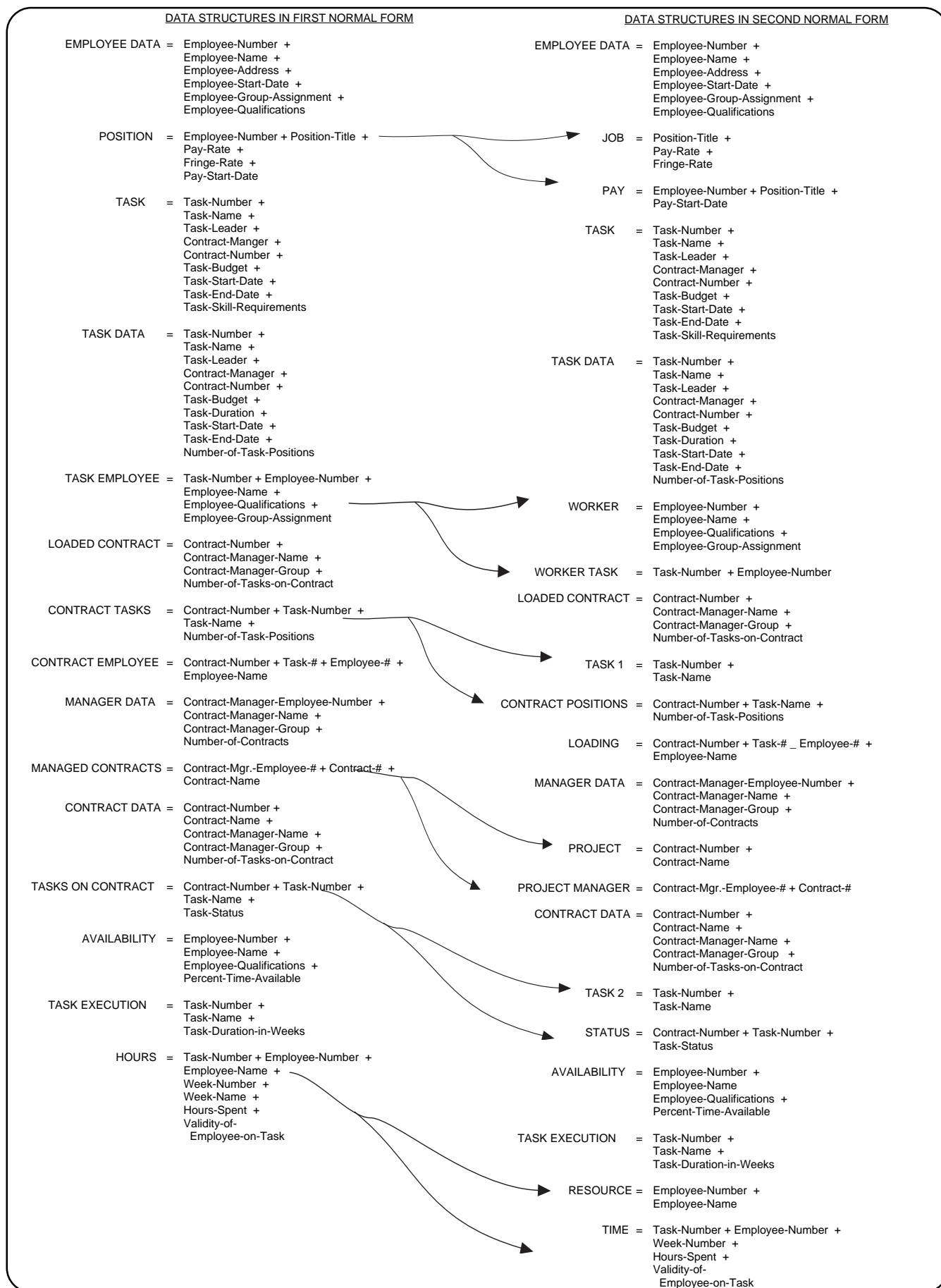


Figure 2.1.7.3.4.f. MSL's Personnel Assignment System Data Structures in Second Normal Form.

CONTRACT TASKS (Contract Number + Task Number + Task Name + Number of Task Positions)

9-62	9-62-01	Do Analysis	3
9-62	9-62-02	Evaluate Results	2
9-62	9-62-03	Write Report	1
9-75	9-75-01	Write Code	2
9-75	9-75-02	Test Code	1
9-81	9-81-01	Write Report	1

TASK 1 (Task Number + Task Name)

9-62-01	Do Analysis
9-62-02	Evaluate Results
9-62-03	Write Report
9-75-01	Write Code
9-75-02	Test Code
9-81-01	Write Report

CONTRACT POSITIONS (Contract Number + Task Number + Number Task Positions)

9-62	9-62-01	3
9-62	9-62-02	2
9-62	9-62-03	1
9-75	9-75-01	2
9-75	9-75-02	1
9-81	9-81-01	1

Figure 2.1.7.3.4.g. *Components of MSL personnel assignment data structures in the second normal form.*

DATA STRUCTURES AFTER COMBINING COMMON ELEMENTS

EMPLOYEE = **Employee-Number** +
Employee-Name +
Employee-Address +
Employee-Start-Date +
Employee-Group-Assignment +
Employee-Qualifications +
Percent-Time-Available

TASK = **Task-Number** +
Task-Name +
Task-Leader +
Contract-Manager +
Contract-Number +
Task-Budget +
Task-Start-Date +
Task-End-Date +
Task-Skill-Requirements +
Number-of-Task-Positions +

CONTRACT MANAGER = **Contract-Manager-Employee-Number** +
Contract-Number +
Contract-Manager-Name +
Contract-Manager-Group +
Number-of-Tasks-on-Contract +
Number-of-Contracts

WORKER TASK = **Task-Number + Employee-Number**

HOURS = **Task-Number + Employee-Number** +
Employee-Name +
Week-Number +
Week-Name +
Hours-Spent +
Validity-of-Employee-on-Task

Figure 2.1.7.3.4.h. *MSL's Personnel Assignment System Data Structures in Third Normal Form.*

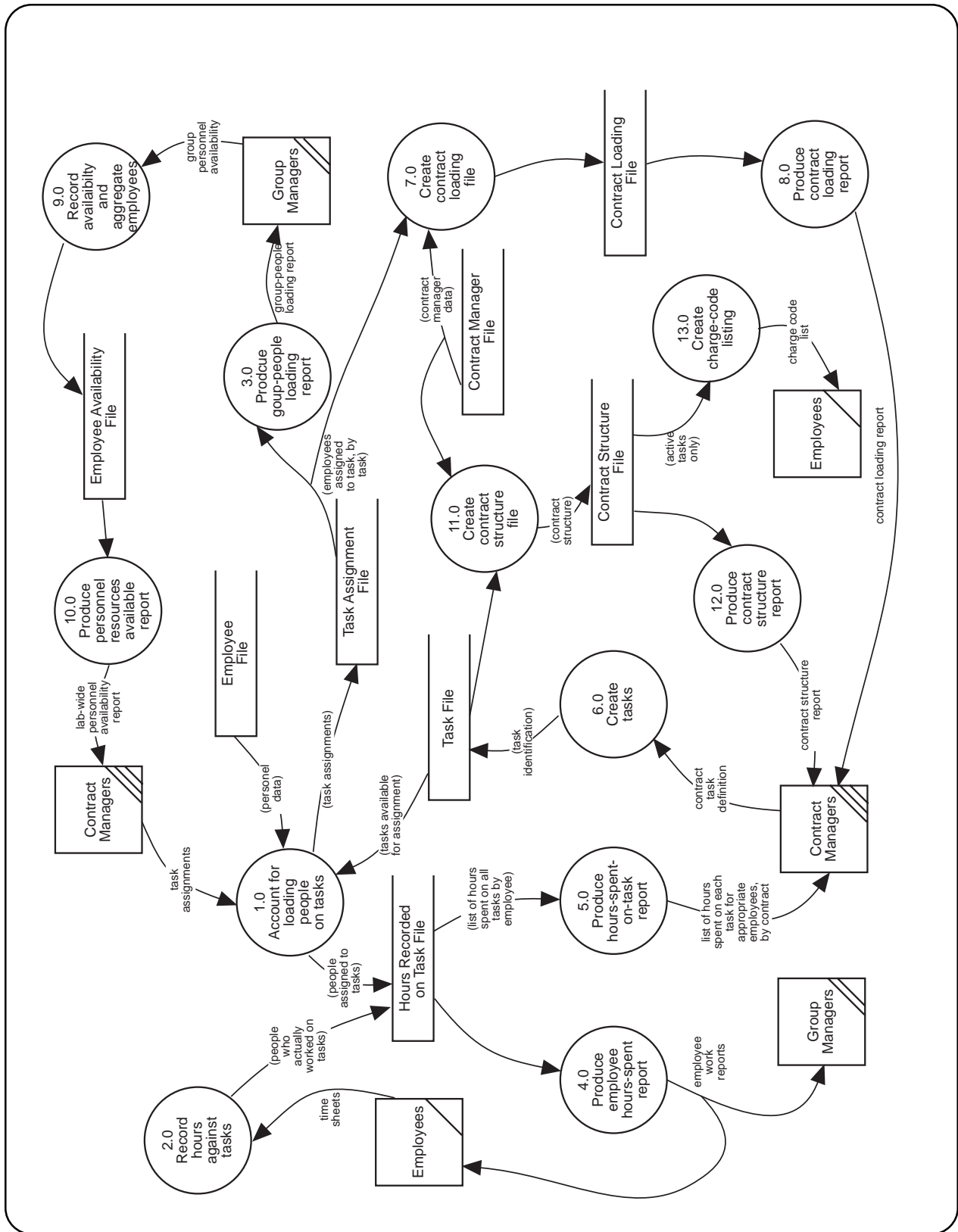


Figure 2.1.7.3.4.i. Revised data flow diagram from MSL's personnel assignment system.

2.0. BUILDING MANAGEMENT TOOLS

2.1. APPROACHES FOR BUILDING TOOLS

2.1.8. LOGICAL INFORMATION ANALYSIS

2.1.8.1. FOCUS ON WHAT YOU MANAGE—WINSLOW HOMER

2.1.8.2. CHOOSING YOUR MANAGEMENT ELEMENT.

If you can focus on the entity you manage toward, your efforts for gathering, storing, and retrieving data will converge rather than diverge.

As we build management tools, the name of the game is to figure out what data we need to put into tools to get the information we need out of the tools. One way to get at the data in the organization for our management tools is to do DFD's and then develop data dictionaries for each information flow. There are other ways to get at the data elements we need. I'll describe another way here. The best way to use depends on the situation. The worst way is to make random guesses of the data you might need.

A management element is the entity you manage toward which your decision making efforts converge. Of all the things that may be in your domain—people, materials, activities, budgets, etc.—something is central. One entity is the hub the others rotate around. If you figure out what the hub is, your decisions and resulting actions and the information and data you use to support your decisions are more focused. Things seem to naturally fit together better. Not knowing your management element isn't a disaster. You can still do your work. However, things aren't so focused and your decision making efforts tend to diverge. You haven't distinguished the hub other entities rotate around.

The time you spend deciphering your management element pays rich dividends. Your thinking and decision making are more focused and streamlined. Your management tools perform better.

I've shown examples of management elements in Figure 2.1.8.2.a. This partial list is only the tip of the iceberg. You'll be able to think of

some entirely different entries and some slightly different entities. Your management element is highly tuned to your domain.

Most of you will see aspects of your domain of responsibility that emphasize more than one, or even all, of the entities in Figure 2.1.8.2.a. You may think you have more than one management element. Indeed you may think all of these are your management elements. I'm asking you to choose one that is more your management element than any of the others. For your management element, you should be able to divide resources and assign accountability, and your management elements should have limits—be discrete.

This isn't easy, but it's worth it. Why? Because if you choose the right management element, information will be focused on your objective; if the wrong element, the information will be scattered and hard to consolidate. That's not a disaster if you're wrong, it's just more difficult.

Some years ago, Virginia Power came to me and wanted help with a methods-type management tool (See Modules 1.4.2.6.3. and 1.5.1.3.4.) called Critical Path Method (CPM). CPM is one of several types of network-based charting techniques used in project management. Any project management book describes CPM. CPM is a precedence diagramming technique showing the precedence of activities in a project. CPM requires that you know the endpoint of the collection of activities. CPM has some advantage over other networking techniques because "precedence is mathematically more precise and allows the

user to exercise much more control over the relationships between activities in the schedule.” (Daniel D. Roman, *Managing Projects: A Systems Approach*, Elsevier, 1986, p. 181.) Notice how CPM focuses on activities, one of the entities in Figure 2.1.8.2.a.

The managers from Virginia Power, who were responsible for producing batches of refueling elements for their nuclear power plants, had recently studied project management at a workshop. They were armed with a tool in search of a problem. My first step was to understand their domain of responsibility.

A difficulty arose in that the managers represented two different endeavor levels: the strategic and the tactical. The strategic decision making in Virginia Power centered around the nuclear fuel cycle and their efforts in maintaining a presence in the mining, milling, conversion, enrichment, and fabrication stages. They had to make make-or-buy decisions for each stage and ensure they had adequate quantities of fuel material at each stage so when they needed to have material at a succeeding stage they didn’t have to wait for feed material. In technical terms, they had a multi-stage, production-to-inventory problem with make-or-buy decisions at each stage.

The costs of waiting one day during the refuelling cycle in a nuclear reactor are astronomical. The corporate policy was to maintain a given quantity of uranium distributed among the inventories at the fuel cycle stages. At that time, uranium was quite expensive and the amount of money tied up in the large total inventory was enormous. (As an aside, keeping large inventories is a way of using the technique of slack resources in managing. If you can’t nail down the process and the cost of failure is high, keep large amounts of slack resources—money, people, material, equipment, energy—available as contingencies. This is the opposite of Just-In-Time.)

The tactical decision making in Virginia Power centered around constructing the reload batch of fuel elements on time to refuel a given reactor during its refueling window.

My recommendation to the managers responsible for the strategic endeavors was that they needed a Material Requirements Planning (MRP) technique. William J. Stephenson defines MRP as “a computer-based information system designed to handle ordering and scheduling of dependent-demand inventories (e.g., raw materials, component parts, and subassemblies).” (William J. Stevenson, *Production/Operations Management*, Richard D. Irwin, Inc, 1990, p. 583.)

What happened in the nuclear fuel cycle and in each of the stages of the fuel cycle was that they did activities that yielded inventories. Of course, they used people and machines to do the activities to get the inventories. To figure out what tool to use and to figure out the focus of decision making, I needed to find the management element.

For the strategic endeavor, the management element was the inventories. In this case, other entities including the people, equipment, and activities aimed at the inventories. Therefore, the people doing the strategic endeavors would get the most benefit from an inventory-oriented management tool, namely MRP.

For the tactical endeavor, the management element was the activities. In this case, the people and equipment played roles in the activities and the inventories came out of the activities. Therefore, the people doing the tactical endeavors would get the most benefit from an activity-oriented management tool, namely CPM.

The managers doing the strategic endeavor could have used the CPM tool they came to me with. If they had, they would have dealt with

inventories as attributes of certain activities. Whether they saw activities as attributes of inventories or inventories as attributes of activities may seem a trivial matter at first glance. But when they selected the tools they needed or they focused the information they used for decision making they'd go the long way if they saw activities as their focal point. Their thinking and effort would diverge. They'd have to search longer in their databases. They'd set up their output formats around activities. They could make activity-oriented decisions easier, quicker, cheaper, and better than inventory-oriented decisions. They'd eventually come upon the answer. I'd rather drive directly toward the answer I need by focusing on the right thing.

The management element is a concept I used to help select or build management tools. CPM and MRP are both methods-type management tools managers use to make decisions with. I quoted Stephenson's definition of MRP not just to define MRP but also to illustrate our common preoccupation with the computer. Both CPM and MRP can be done without the computer. In most cases, the computer helps us do more cumbersome problems quicker. However, if we lose sight of the principles behind the tools, we'll apply them improperly. Remember the caveman story in Module 1.4.5.2.2.

I'll now give you another example of choosing the right management element. In Figure 2.1.8.2.b., you see the classic dilemma for a teacher, who manages the classroom, which is the special student who needs the teacher's

time. The teacher's ready to go to class on time. Two hundred people are waiting on him. If the teacher gives the time to the student, the class suffers. If the teacher scurries off to class, the student suffers. Therefore, the teacher's choice in this dilemma indicates the management element. Does the teacher focus on the student or the class? Possible management elements might be lesson plan (today's lesson), course (e.g., engineering economy), class (e.g., this particular section of engineering economy), the room, the student, a student group (if the students work in groups in that class), and many more. If the class is the management element, each student's name is an attribute of that class, as is the course number, the room number, the meeting days and times, the various lesson plans, and many more. If the student is the management element, the other entities are attributes of the students in the teacher's domain.

The right choice of management element helps you filter need-to-know information from irrelevant data. Information leading from other entities to the management element stand out more readily.

In the Management Systems Laboratories (MSL), project managers see people as resources to be spread over activities. The financial system focuses on transactions as opposed to dates, accounts, or balances.

If you haven't properly defined your management element, you're more likely to suffer from information overload.

CHOOSE YOUR MANAGEMENT ELEMENT FROM A REPRESENTATIVE SAMPLE.

- Inventory - a defined accumulation of a resource
- Activity - a timed effort to achieve a goal under single direction
- Transaction - a transfer of responsibility
- Budget - a comprehensive and specific plan for using resources
- Person - an individual
- Milestone - an important point in time

Figure 2.1.8.2.a. *What's your management element?*

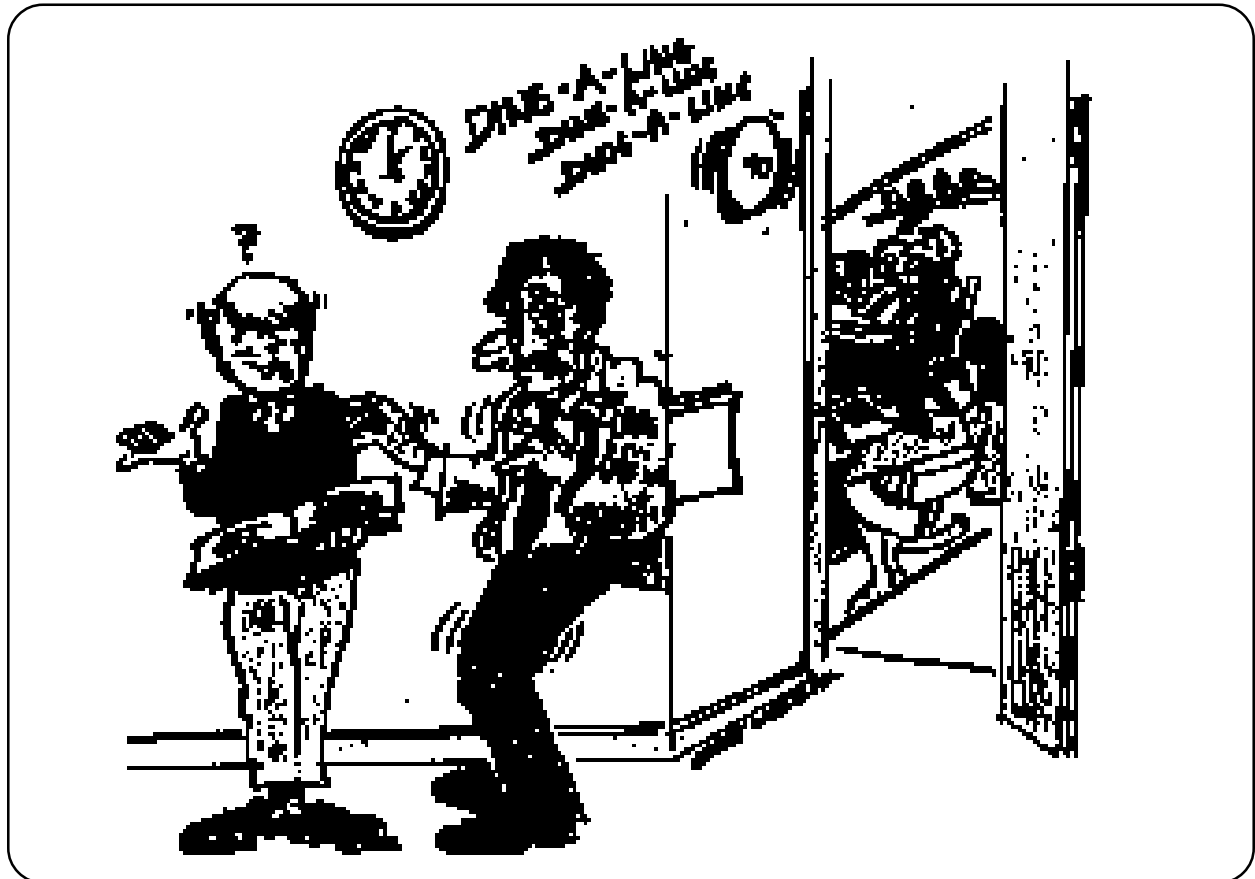


Figure 2.1.8.2.b. *“That answer’s right—huh?”*

2.1.8.3. CHARACTERIZING AND ACCESSING YOUR MANAGEMENT ELEMENT.

Another way to develop a list of data elements is to characterize your management element.

Attributes are the characteristics of your management element, and attributes lead to data elements.

In Figure 2.1.8.3.a., I've grouped attributes into five categories. The categories include the ideas of resources, schedule, and quality (or results) as well as relationships between your management elements and a catch-all I call identification. You saw the ideas of resources, schedule, and quality before when I discussed the project management pyramid.

The data-flow-diagram perspective of data elements works best when we know both ends of the information flows as in more certain environments. The management element perspective is best for more uncertain environments.

Each category in Figure 2.1.8.3.a. can be subdivided. The data elements appear when the attribute categories cannot be divided further.

Data are characteristics. When we use a reference point, we make information, and we include bias. Three of us may have the same characteristic. As shown in Figure 2.1.8.3.b., I'm persistent, you're stubborn, and he's pig-headed. The interpretation depends on the bias. Other examples of different perspectives of the same characteristic include: 1) consistent, predictable, and dull (boring), and 2) judicious, neutral, and indecisive.

Remember that we're looking at an abstract thing from several different perspectives to try to come to grips with it. When you do data flow diagrams, the natural progression is to identify information flows and determine the data elements that make up the flow. This technique

gives you one list. Now we're looking at management elements and determining characteristics. This technique gives you another list. The lists won't be the same. However, one isn't better than the other. We need both to generate a complete list.

What we have with data elements is the ability to know what slots need to be filled with data and the appropriate data to fill those slots. We put values for the data elements into databases to retrieve when we want to make information.

Some attributes can be grouped into structures for accessing, monitoring, and controlling as shown in Figure 2.1.8.3.c. Attributes can continue simply to describe the management element or they may be elevated and structure the environment of the management element.

This elevation from attribute to structure is your choice but should respond to a need for greater accessibility or easier reporting, updating, and maintaining of data. The penalty for elevating an attribute is a more complex information system.

Consider an Hollarith card. (You may not remember Hollarith cards—the cards we punched to input data into and sometimes to output data from a computer.) The monitoring element is in the first field and attributes are in the following fields. The monitoring element might be last name or social security number.

Here's a story for understanding how to recognize environmental structures for characterizing and assessing a management element. In Figure 2.1.8.3.d., a newly married student has come to the departmental office in our univer-

sity to have her marital status changed on her records. My department keeps student files by name. Someone asks the question, “Which students are married?” After looking through the files enough times, we mark married student files with red tape. Then someone asks the question, “Which students live in Virginia?” After looking through the files enough times, we mark Virginia-student files with blue tape. Now if someone asks the question “How many married students live in Virginia?”, the answer is easy to get. We look for files with both red and blue tape. What I’ve described is a very simple instance of a modern database technique that allows two environmental structures.

We call the matrix in Figure 2.1.8.3.e. a schema, which is a diagrammatic presentation of your domain of responsibility. From this schema, database structures, information flows, and interaction points become clear—either as they exist or as they should exist. An example of an interaction point is whose office is next to whose office.

A primary purpose of the schema is to identify existing or potential gaps in responsibility or coordination. The schema is the starting point for defining data elements and information flows. Each axis on the matrix is a path into the data characterizing the management element.

It’s said that everyone wants to get to heaven, but nobody wants to die. You have begun to pay the price to understand what you manage well enough to fix automatic procedures according to a set structure. You know the phrase “Getting there is half of the fun.” In this case getting there is no fun at all. Certain things are better to have done than to do.

Think of one of the information documents

you’d like to support one of your decisions. Whether it be text, checklist, table, or graphic, identify each data element on the document. You may have overlooked data elements. Any alpha-numeric unit on your document constitutes a data element. Ultimately we want to define data to access, store, and retrieve. Identifying the data elements in all the information documents for all the decisions is the other way to generate a representative list.

Relate the attributes of your management element to the data elements you identified on your document.

For any data element you must know the following:

- Who obtains the data? (Name of person)
- What is their source? (Name of agency or person)
- Who verifies the data? (Name of person)
- How frequently should the data be updated? (hours, days, weeks, months, etc.)

Data are expensive to maintain. You must identify the important pieces of data. Figure 2.1.8.3.f. shows a case of determining what should be measured to make data.

Two farmers with contiguous farms had horses they couldn’t tell apart. First, one of the farmers bobbed the tail of his horse. The other horse ran into a barbed wire fence and pulled off his tail so that again the farmers couldn’t tell the horses apart. The first farmer then notched the left ear of his horse. The other horse ran into the same fence and notched his same ear. Then the farmers decided to determine if the horses are of different height. Wouldn’t you know it, the black horse was two hands taller than the white horse.

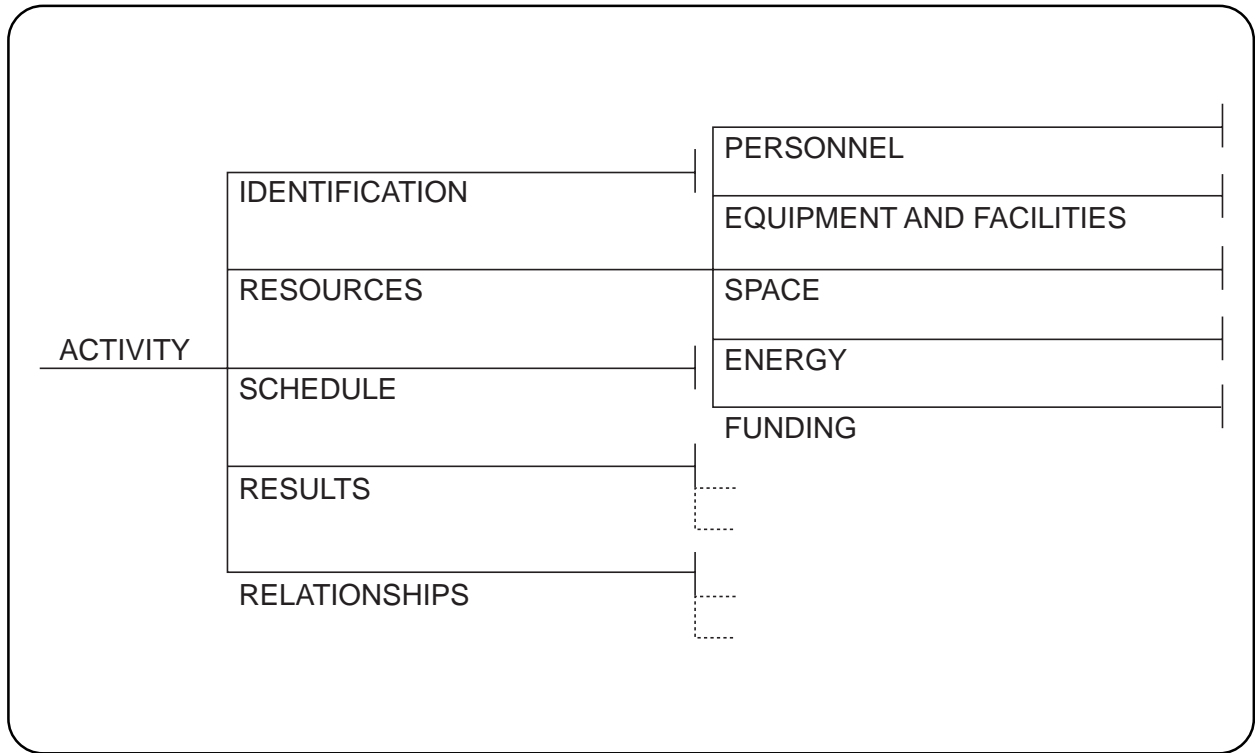


Figure 2.1.8.3.a. Five attribute categories describe an activity.

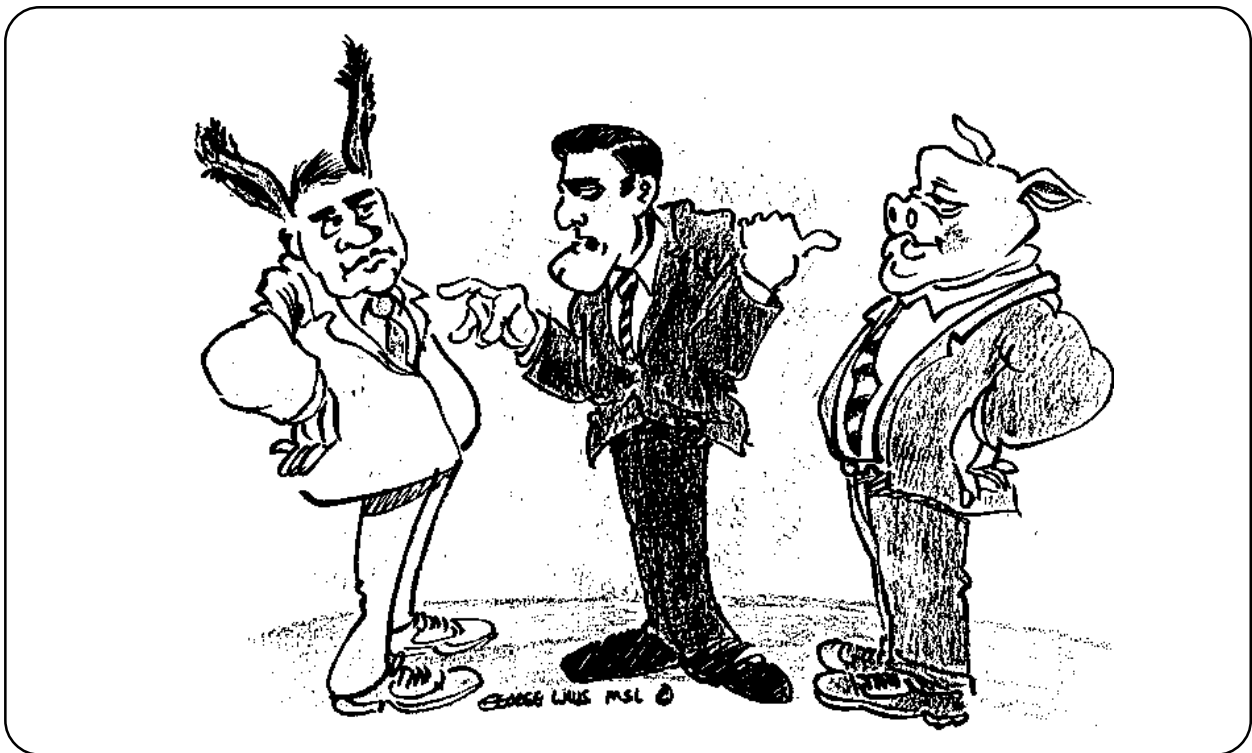


Figure 2.1.8.3.b. “I’m persistent, you’re stubborn, and he’s pigheaded.”

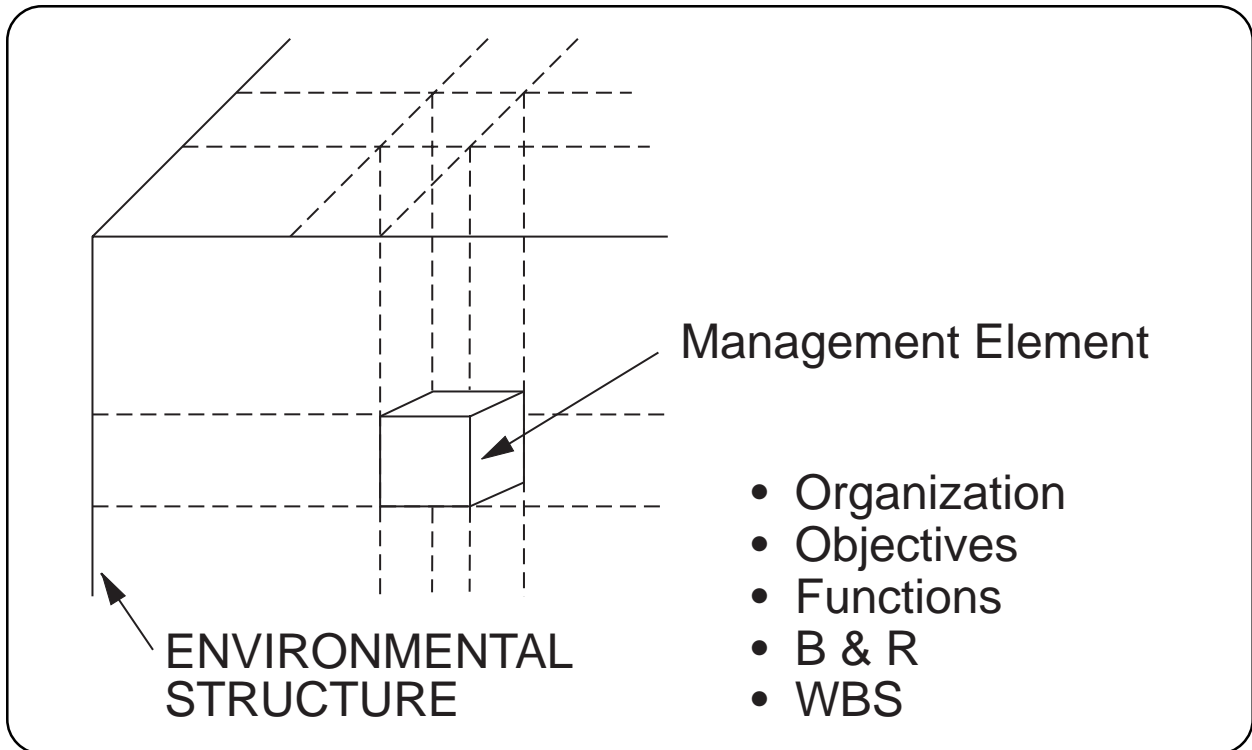


Figure 2.1.8.3.c. Environmental structures allow ease of accessing, monitoring, and controlling information.

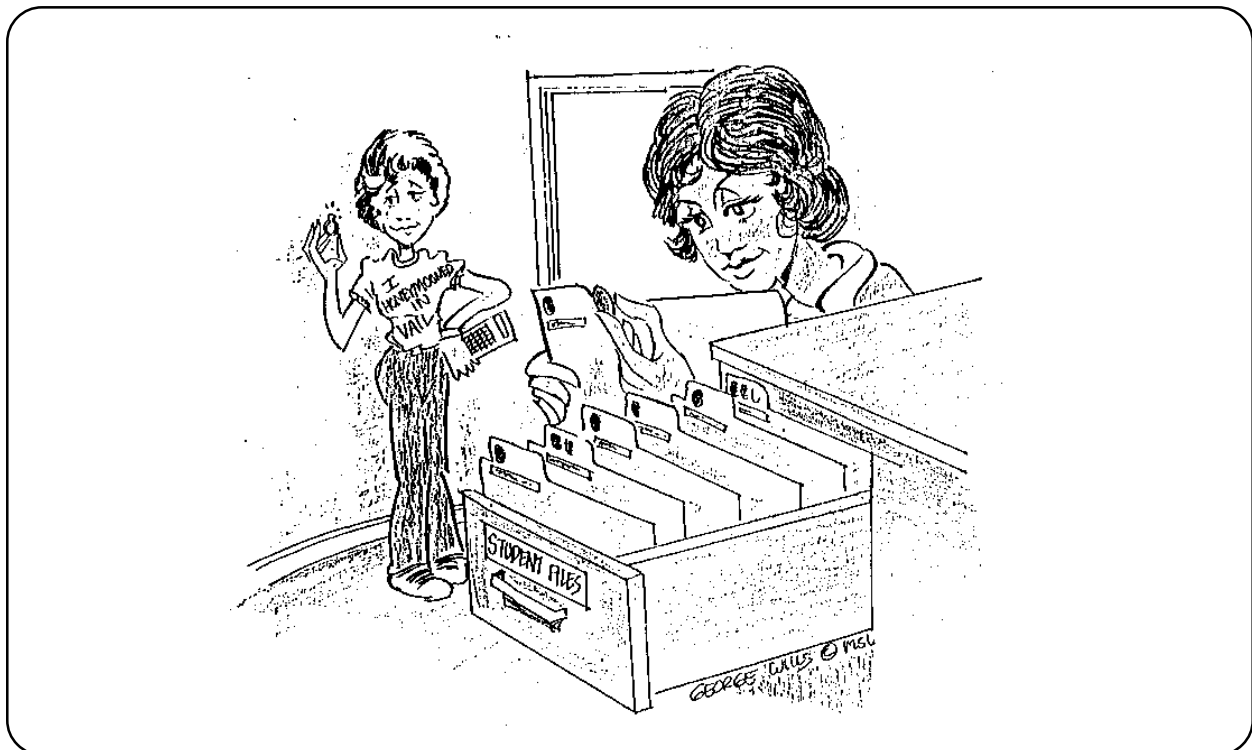


Figure 2.1.8.3.d. “For the record, change my records.”

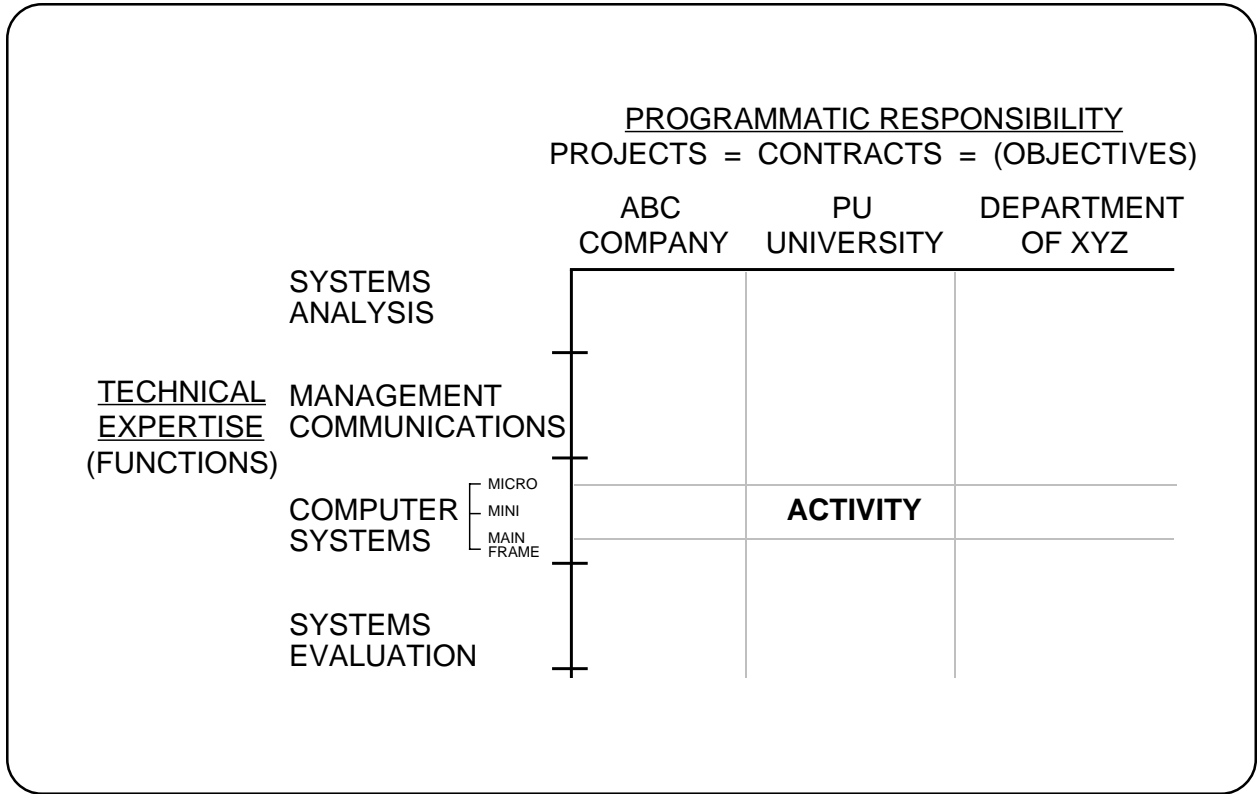


Figure 2.1.8.3.e. “MSL manages activities in a project-management matrix organization.”

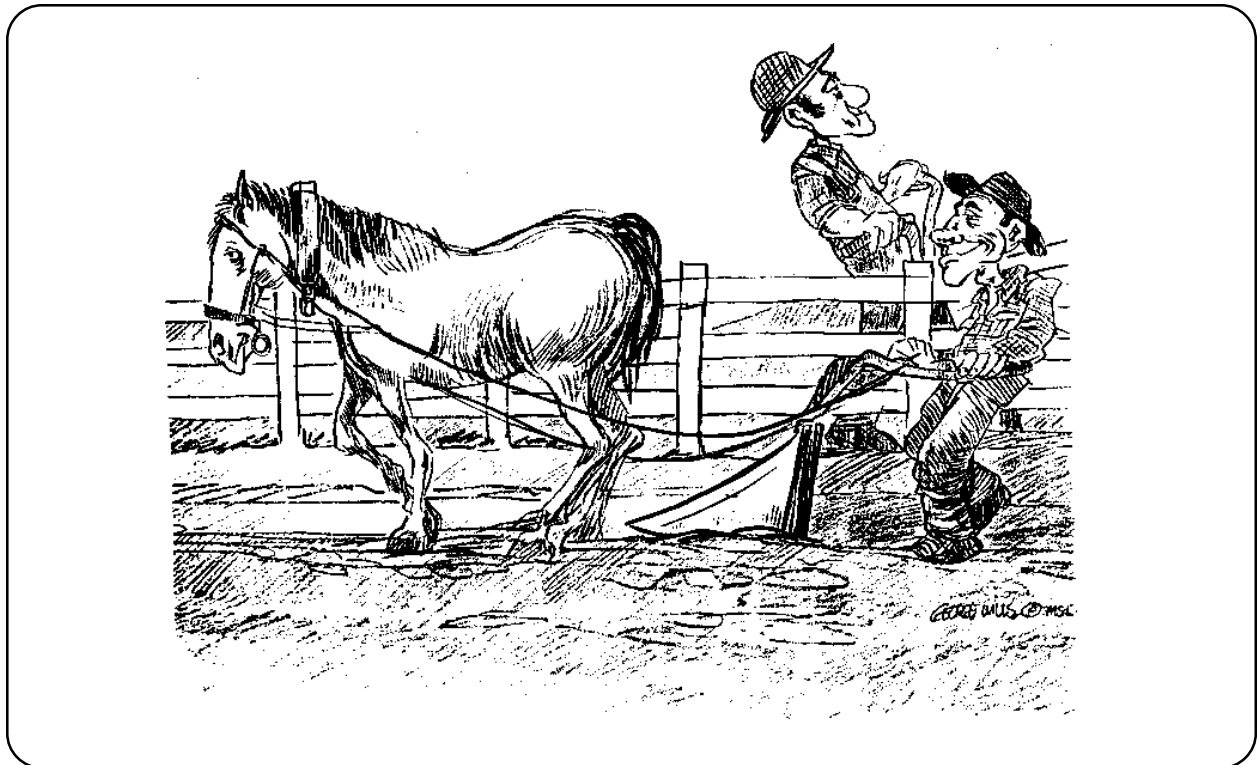


Figure 2.1.8.3.f. “Whose horse is whose, Clem?”

