



---

## **2.0. BUILDING MANAGEMENT TOOLS**

### **2.1. APPROACHES FOR BUILDING TOOLS**

---

**2.1.1. MODELING—PIETER BRUEGEL.**

---



## 2.1.2. THE DATA-TO-INFORMATION CHAIN.

**Since all management tools convert data to information, you follow much the same procedure and use similar tools, guides, and skills in building the tools .**

As an industrial engineer, you'll likely be asked to produce a management tool, whether it be an organization structure or a management information system. To do so, you'll need three things: 1) existing tools and guides, 2) a procedure to tie tools and guides together and to build such a tool, and 3) the systems analysis skills to use the tools as methods and carry out the procedure. You'll either have the resources and the authority to build the tool within your domain or you'll have to contract out to have someone else build the tool for you. In either case, you'll want to be familiar with the needed procedure and skills.

We've seen the Management System Model (MSM). It shows us what our management system looks like and how it works. Once we've delimited a domain of responsibility, we can use the MSM to more clearly probe into what makes up that domain.

Any domain has management tools, although they may not be very sophisticated or structured. They may only be in the heads of a few people in the organization and not consistently or widely used. For example, the real organization structure (one of the management tools) may not be written down but instead in someone's head. The marketing plan may be in someone else's head. The plan really exists, but you have to ask the person who knows to find out what the marketing plan really is. Best of all, if you ask the right person, you'll get the most up-to-date version of it. However, it's hard to get everyone to sing off the same sheet of music if we have no sheets of music to sing off of. So, we opt to write down some version of the plan and let everyone read the same,

albeit usually out of date, information.

The same thing is true with the data-to-information chain, or MIS (Management Information System). It may be in someone's head. The marketing *plan* is usually best found in one person's head, but the latest marketing *data* are in a whole bunch of people's heads, which is even more reason to write the data down. Where do we write them? On paper, the back of an envelope, in a notebook, on cards, in a computer, or wherever. You'll find some pretty primitive MIS out there.

Let's take a moment to recall what the MIS is. It's the data-to-information chain that routinely converts data to information. The data-to-information chain is special, and different from other management tools, because it most frequently and regularly acquires, stores, retrieves, and manipulates data to compare to reference points (biases) to make and display information.

Figure 2.1.2. shows the data-to-information chain and emphasizes the links we use to get data and step them through a routine process to provide information for decision making. Notice the parallel between this figure and the MSM. The operation is *what is managed*. I'll use those terms interchangeably. Management intelligence represents the *who manages*. The point of the figure is to show the details of one of the management tools in *what is used to manage*.

All the tools convert data into information. The one in Figure 2.1.2., the MIS, converts data frequently and routinely. Because of the

amount of data, the frequency and speed we need to convert them, and the varying rates at which different data become out of date, we'll focus (1) on the steps (links of the chain) and substeps of the conversion process and (2) on the steps we use to figure out what tool to build and to design, build, implement, evaluate, and modify that tool.

Each link in the chain represents a type of technical expertise we need in building and operating an MIS. The links you see full-on represent things we have to build (emphasizing nouns) and the links you see on-side represent what we have to do with data or information (emphasizing verbs) to get to the storage or the programs, etc. As an industrial engineer, these links should intrigue you. You're trained to solve general problems like these. Shouldn't inventory analysis apply to data storage? We have fast and slow movers in data. Data also have shelf lives. Should materials management apply to acquiring and retrieving data? How about human factors and information portrayal?

For links of the data-to-information chain, I use terminology typical of MIS and especially computer-based MIS. I believe we could generalize these links and apply them, or groups of links or sublinks, to any type of management tool that converts data to information.

Whether in a primitive form, like someone's head, a clipboard, or a card file, or in a sophisticated form, like computerized data bases, we're sorely tempted to think most about how good a new MIS could be. We should really first think about how good the existing MIS is.

Put this firmly in your mind. When you look at a management tool, start by considering what's right with the existing tool, not what's wrong with it. Remember the manager, or user, has a management system. They may not be able to tell you about components and

interfaces, but components and interfaces are there. And the management tools are there and they work (for better or worse) in relation to *who manages* and to *what is managed*. The tools are what they are for a reason and they match the interfaces about as well as can be done without anyone realizing the interfaces are there to be matched. So there's really more right with the tools in regard to the MSM than there is wrong. If you don't figure out what's right, you'll probably build a tool that fixes a lot of what was wrong with the old tool but does little of what was right with the old tool (neglecting dumb luck, of course). We call what you've built a failure.

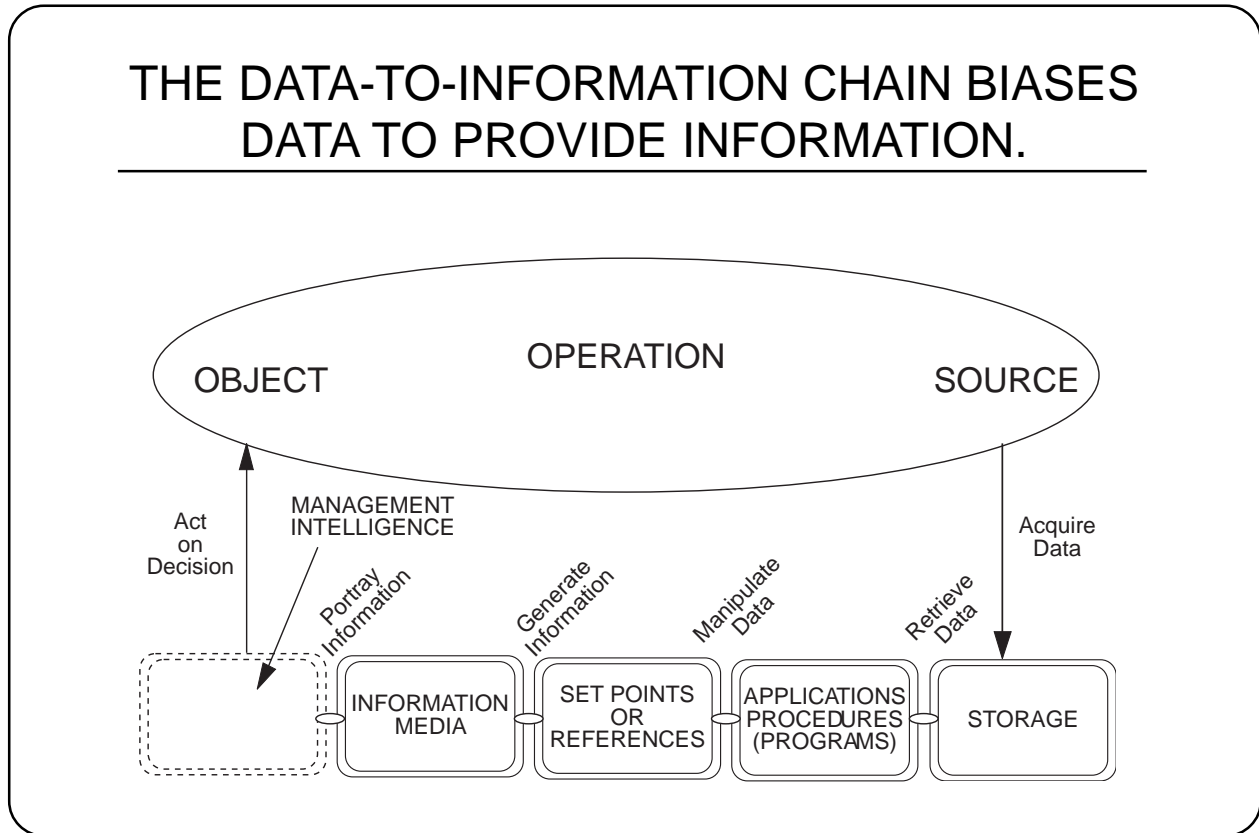
So, we can figure out what we have in the domain of responsibility by using what we know about the MSM and the various ways we can characterize the management system (like the pursuits, endeavors, and so on). We also know we start by looking for what's right with the existing tools. But where are we to be when we build a new tool and how do we get there from here? (By the way, I've just stated the engineering method in its simplest form: know where you are, know where you want to be, and figure out how to get from here to there.) We can solve any problem by doing those three things. However we don't have to do the first two in that order. Also, it's hard to do the third without doing the first two. How many people try to do the third step too soon?

We can figure out where we want to be in designing management tools by using management system analysis (MSA) to logically figure out what data the management tools should be accessing based on what information those tools should be portraying to support decision-making. We can also use something called the automation objectives model I'll describe later.

What I'm going to do now is concentrate on a procedure to use for getting from here to there

in management tool building. I'll talk about this procedure now for two reasons. First, books on MIS focus on such a procedure. Second, implementing the procedure requires a set of skills all industrial engineers should have. I want to focus much of this book on those skills. [By the way, industrial engineers should have these skills for building any kind

of management tool. Remember plans, quantitative models, and the like convert data to information too. They're just not as repetitive, frequent, and routine as the MIS. For that matter, industrial engineers should have these skills for solving any problem—or better yet, for figuring out what problem they have to solve and then solving it.



**Figure 2.1.2.** *The data-to-information chain biases data to provide information.*





---

## **2.0. BUILDING MANAGEMENT TOOLS**

### **2.1. APPROACHES FOR BUILDING TOOLS**

#### **2.1.3. THE SYSTEM LIFE CYCLE**

---

### 2.1.3.1. THE SYSTEM LIFE CYCLE

---

#### **The steps and stages for developing any management tool flow logically.**

The management tool building procedure is easier to outline and describe in terms of the MIS. So I'll focus on the MIS in this discussion. We can look at this procedure from many perspectives. Each perspective shows us something different. Figure 1.1.20.1.3. is a control-oriented view highlighting activities and decisions. Building an MIS is like building anything else—it's a project. Project management skills are needed—but more of that later. A control-oriented diagram is logical—call it a logic diagram. It shows decision points and branching. (Figure 1.1.20.1.3. is so general you don't see much branching; but, of course, each decision in the figure has more than one possible outcome.)

Figure 2.1.3.1. highlights the work flow in building an MIS. The process flow diagram in Figure 2.1.3.1. is similar to the logic diagram for the system life cycle in Figure 1.1.20.1.3. The process flow diagram neglects the decisions between the activities. Figure 1.1.20.1.3. emphasizes the logic in building a management tool and the frequent management input and decision needed in building the tool. Figure 2.1.3.1. emphasizes the value of prototypes the difference between the working MIS and the goal, and the universality of evaluation and documentation.

Figures 1.1.20.1.1.a. and 1.1.20.1.1.b. focus on the process of building an MIS. They focus on the information and the information-conversion processes (for example, in Figures 1.1.20.1.1.a. and 1.1.20.1.1.b. the structured analysis process converts the feasibility document into structured specifications.) Figures 1.1.20.1.1.a. and 1.1.20.1.1.b. emphasize information conversion processes in the rounded rectangles and information flows on the arrows. The processes in Figures 1.1.20.1.1.a.

and 1.1.20.1.1.b. are the activities in Figure 1.1.20.1.3. We've talked about the different systems analysis tools of logic diagrams and information flow diagrams. Information flow charting, like what we see in Figures 1.1.20.1.1.a. and 1.1.20.1.1.b., is especially suited for viewing what we want to do when we build a management tool or especially an MIS. You'll see a lot more about information flow charting in this course. That's another systems analysis skill.

Figures 1.1.20.1.3., 2.1.3., and 1.1.20.1.1.a. and 1.1.20.1.1.b. represent three ways of making pictorial representations of what we need to do to build a management tool. The logic diagram in Figure 1.1.20.1.3. looks at decision points, branching from decisions, and the actions getting us from one decision to the next. The work flow diagram in Figures 2.1.3.1. shows us the steps we take in an effort, the sequence of those steps, and the outcomes from the steps. The information flow diagram in Figures 1.1.20.1.1.a. and 1.1.20.1.1.b. highlights the information flows and information conversion processes.

Engineers like pictorial representations because engineers think logically and in sequential steps and they're used to dealing with more tangible or quantitative presentations of their ideas and their results. You can use these three different pictorial representations for almost anything. In Figures 1.1.20.1.3., 2.1.3.1., and 1.1.20.1.1.a. and 1.1.20.1.1.b., we're looking at building the MIS. Later we'll use these three representations to look at the domain of responsibility of the user—a domain we're building the MIS to serve. As a matter of fact, that's exactly what we'll do in one of the early steps in the procedure for building the MIS. We'll do at least information flow diagrams for the

domain we're building the MIS for.

Let's now look more closely at the steps of the procedure in Figure 2.1.3.1. We commonly use to build an MIS. The analysis steps and implementation steps are the responsibility of the manager using the resulting management tool. Also, they're historically the weakest steps, and probably the steps most responsible for the 70% failure rate in management information systems. Note that the manager plays a crucial role in building the management tool.

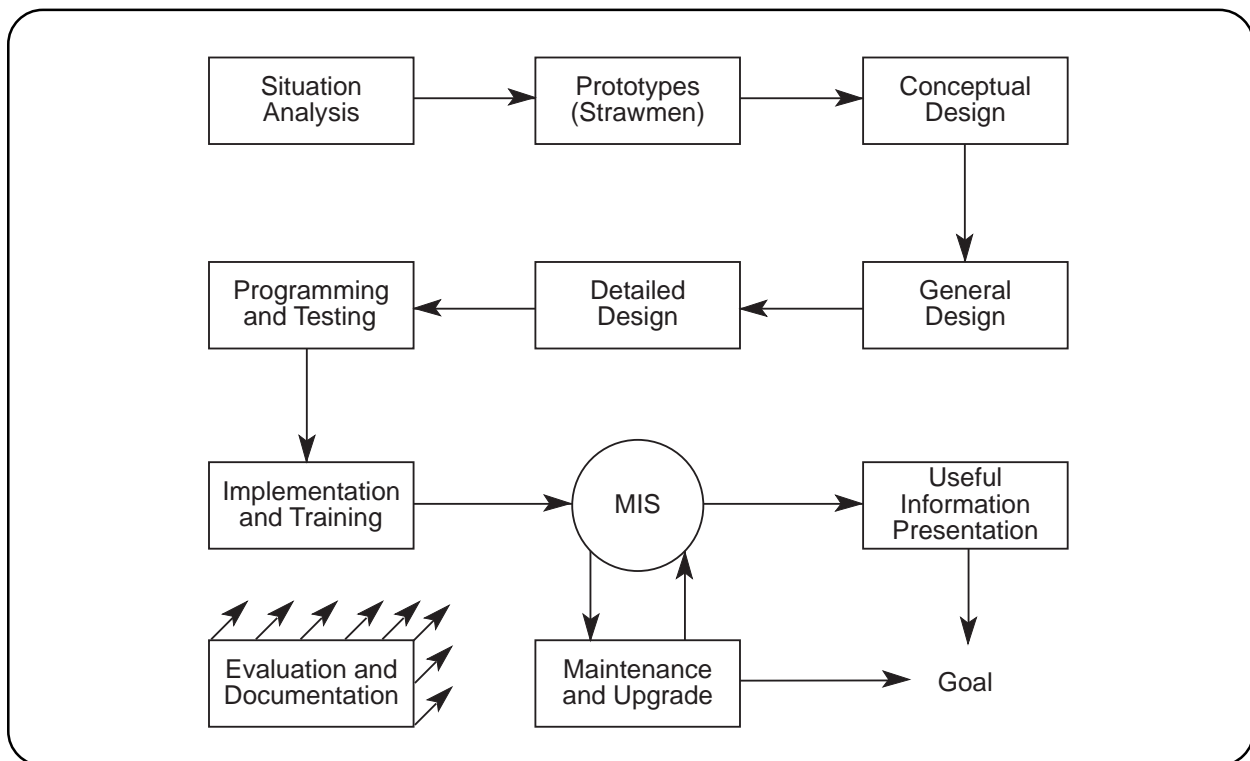
Two circumstances which lead to the high failure rate are (1) an inaccurate analysis of the management situation, usually from a failure to *follow-through* causes an inappropriate MIS design, and (2) an appropriate MIS is not *followed up*—monitored, maintained and enhanced to ensure an ongoing fit to the information requirements of management.

The first pitfall is avoided by careful attention to the analysis steps in the process flow dia-

gram. Out of these steps will come a clear picture of management's information needs. Once these needs are accepted and documented, the situation should not be fought. In the military, we learn "don't fight the situation." Once you understand the situation for what it is, work with it—don't wish it away or assume the situation is something it isn't. This adage applies to management tool building—the MIS must be designed *to complement the situation*.

The second pitfall is avoided by insisting on a system *follow-up*. This is achieved by following up the implementation steps of the process flow diagram as shown in Figure 1.1.20.1.1.b.

The design steps are for the tool builder. The tool builder must be good at getting information from the user out of the analysis steps, putting that information into practice in the design steps, and then handing off the results to the user so the user can be successful in the implementation steps.



**Figure 2.1.3.1.** The process flow diagram includes the user.

### 2.1.3.2. AN OVERVIEW OF THE NINE STEPS

---

**The nine steps in the system development life cycle highlight where you'll use your system analysis system.**

Note the graphic portrayal of the Implementation Procedure Diagram in Figure 2.1.3.1. The first three steps, combining to form the Analysis Stage, yield a conceptual design which defines the objective(s), and serves as a monitor on system design. The Design Stage, which follows, requires special training and experience—it almost always includes the assistance of automation specialists.

The MIS is actually a complete system following the seventh step: Implementation and Training. The goal, however, hasn't been reached. The goal is to have an ongoing flow of appropriate action based on appropriate information. In terms of the Implementation Procedure Diagram, the goal is reached only after the system has been in place, been maintained and upgraded, and used to present useful information.

Two more steps, the eighth and ninth—Maintenance and Upgrade; and Useful Information Presentation—are probably the most overlooked. They are absolutely essential, however, to the success of the system.

The most difficult and the most important steps don't belong to one stage and don't follow one or several of the nine steps. These important steps are documentation and evaluation. We all have experience with computer systems, plans, time management techniques, and other tools we can't use very well because the documentation is poor. In these cases, we're concerned with user documentation. In

truth, poor user documentation, operational problems in the system, and improper fit can usually be traced to poor documentation during one of the steps in the Implementation Procedure Diagram. A correct, clear, concise, comprehensive information requirements document resulting from the situation analysis step helps us ensure a good fit. Design documents at the several design steps help initiate operational problems. In short, we need documentation at each and every step in the Implementation Procedure Diagram. These documents are crucial for communication—communication between users and analysts and between analysts and tool builders.

Evaluation is probably the most neglected step of all. Many people don't like conducting evaluations because they don't like being evaluated themselves. Evaluation must start during the first step and continue throughout all the steps of the Implementation Procedure Diagram.

System development seldom proceeds in a purely sequential fashion. Once there is agreement on an integrated design of the overall system (following the Conceptual Design Step), the remaining steps are often applied incrementally to subsystems of that overall design.

With these considerations in mind, we can now turn to the nine steps of development in the Implementation Procedure Diagram.



### 2.1.3.3. THE ANALYSIS STAGE

---

**The analysis stage is most important because it's the foundation for the following stages.**

Although the Implementation Procedure Diagram has the built-in dual-path feature, it is best understood by way of a close look at its individual steps, and the role those steps play in the three stages of systems development.

The first three steps of the Implementation Procedure Diagram comprise the Analysis Stage (Figure 2.1.3.1.). These steps are 1) Situation Analysis, 2) Prototype Development, and 3) Conceptual Design.

Situation Analysis is the most important step because it ensures the right approach to the right problem. Remember: once the situation is understood and management's needs are defined, the system must be built to cater to that situation. In other words, change the system before you change the situation. This step *belongs to the manager*—it is where the goals, strategies, and priorities of the system are defined.

This step is critical, so I'll divide it into seven substeps. These substeps are a) define the domain of responsibility; b) develop management issues, and define the *Critical Success Factors*; c) understand the existing system; d) determine information needs; e) uncover gaps and overlaps; f) prioritize needs in terms of consequence and immediacy; and g) identify the potential for creating prototypes.

The second substep defined above, the development of management issues, is much studied and reported; many methods are used for determining information needs.

#### **Critical Success Factors (CSF)**

These are factors which management deems necessary to the success of their organization. The most common CSF's would include cost structure, product quality and innovation, customer satisfaction, management development, and any change in corporate culture and attitudes. Note that four of these common CSF's are soft data—they are intangible entities which cannot be quantified.

In the third substep of Situation Analysis, you should evaluate existing management information capabilities. Many of these capabilities will find their way into the implemented design—in a much more integrated manner.

The second step in the Analysis Stage is Prototype Development. Prototypes (also referred to in Figure 2.1.3.1. as “strawmen”) are small preliminary subsystems developed to meet immediate needs and allow critical feedback. Prototypes pay for themselves through immediate application; they pay again because they help clarify the architecture of the final system. Often a manager needs something concrete to look at; this step provides just that. This step is borrowed from the bottom-up approach to systems development; the other two analysis steps are taken from the top-down.

The third and final step in the Analysis Stage is the Conceptual Design. This step ensures that overall system requirements are considered before specific, narrow capabilities are developed. The necessities of the system are

defined here: 1) the system goals; 2) the functions that must be performed to meet those goals; and 3) the information required to carry out those functions.

Following the completion of the Conceptual Design step, a document should be presented

to management for approval and establishment of development priorities. This document might be called the Systems Concept Document. It should contain as much detail as possible—the tighter the specifications, the smaller the likelihood that the designed system will stray from them.

### 2.1.3.4. THE DESIGN STAGE

---

**In the design stage you determine how you're going to meet the situation to satisfy the information requirements of the decision maker.**

Steps four through six of the Implementation Procedure Diagram comprise the Design Stage. These steps are: 4) General Design, 5) Detailed Design, and 6) Programming and Testing. Generally, this stage is of least importance to the manager, since much of its execution will be carried out by automation specialists. As we emphasized before, tighter specifications in the Analysis Stage will make a big difference here. Still, a manager must understand the components of this stage, since its output will have a huge bearing on the final system.

The General Design step initiates the System Design Stage. With the system concept defined (and documented), General Design again focuses on user needs. Flow charts are developed to illustrate system processes (not computer processes, but the flow of activity—people's procedures and interfaces). I've defined user issues in a manner similar to that used for resolving management issues in the Situation Analysis step.

Later in the General Design step, important relationships between the subsystems must be determined. This holds true for both manual systems and automated systems. Inputs, outputs, and all files necessary to the system are identified here. System output formats are very important and historically have been neglected.

Following this, gross estimates of the size of the system have to be made. As a result of these estimates, and other management considerations, computer hardware and software needs are defined; most importantly, development priorities are established.

This design procedure focuses on information flows in and out of a mode. The General Design step results in a General System Specification (which, needless to say, ought to be a formal system document) which serves as the functional baseline against which system capabilities can be judged.

With the user requirements well established, the Detailed Design step focuses on the technical development of a system to meet those requirements. User-related processes are translated into specific programs with detailed performance and test specifications. Data bases are designed here and input and output formats are completed.

All technical issues must be identified and resolved in this step. Following this, computer hardware and software requirements are specific. The Detailed Design step results in the development of the Detailed System Specification, which will serve as the technical baseline for all ensuing development and implementation.

In the Programming and Testing step, computer resources are acquired, computer programs are written from the detailed design, and documented used procedures are developed and tested. Programs are prepared in accordance with approved standards and conventions. Unit tests must be performed here, and the operational data bases are loaded. Test plans and acceptance specifications are established in this step.

After establishing acceptance criteria, we conduct system and subsystem testing of both computer hardware and software. Successful

execution of the Programming and Testing step results in documented computer programs, programmers' guides, input screens, output formats, and documented test procedures and criteria.

In discussing the design steps, I talked about

data bases, input screens, hardware, and software. If the MIS is not computer-based, I'd talk about file cabinets, data gathering forms, rolodexes, and procedures for gathering, storing, and updating the data kept in the manual system.

### 2.1.3.5. THE IMPLEMENTATION STAGE

---

**In the implementation stage you find out how well you meet the information requirements.**

The final three steps (seven through nine) of the Implementation Procedure Diagram comprise the Implementation Stage. These steps are: 7) Implementation and Training; 8) Useful Information Presentation; and 9) Maintenance and Upgrade.

The Implementation and Training step is a transition to the new system. Users must be trained here, and the old equipment and procedures are replaced with the new. This step is accomplished through the integration of tested systems procedures and programs, documented human procedures, and trained people, into a cohesive people machine system.

User manuals and training sessions are designed in this step. Procedures are established to back up system use and protect the user against loss. Often, implementation is applied incrementally, to ensure proper integration before the entire system is activated.

The result of this step is an installed management information system, but the work is far from done—the following two steps are critical to the success of the system; historically they have been the most overlooked.

The Useful Information Presentation step takes into account a number of important considerations about the nature of information. Information is *biased data*; as such, it should be biased to meet the needs of the system user (in this case, the manager(s) who retested the system).

To satisfy the cognitive style of the user, the

presentation of information must match information *portrayal* (whether it be graphic, table, checklist, or text) and *perception* (the value the user places on the data and information). The interface between information portrayal and perception is a critical point of resolution—the result should be that the manager is sufficiently comfortable and confident in the system to guarantee the system's integration into the decision-making process. As a tool builder, you should try to secure some sort of formal declaration of this confidence.

Although the system is now up, running, and used, the possible pitfalls are many. This is where the Maintenance and upgrade step comes in. To be useful for any length of time, the system must be continually upgraded to include any needed improvements as the user matures in using the system the manager's operation changes and evolves. Additionally, the system must be maintained to correct for bugs that surface through continued use.

Periodically, the system should be evaluated. This is usually a difficult process; nobody likes to be evaluated nor to have his or her system brought under scrutiny. Nonetheless, evaluations will ensure the system fits the changing needs of management, personnel, and any auxiliary organizations involved with the system. Additionally, a system operator should be designated, and the procedures for operation should be defined and documented.

The result of this final step, which actually continues through the life of the system (Figure 2.1.3.1.), is a system which works to meet its goal.



### **2.1.3.6. THE FOLLOW-UP STAGE**

---

**A procedure is necessary but not sufficient.**

The follow-up, emphasized so heavily through the steps in the Implementation Stage, is of paramount importance. Many systems fail because managers believe the work is over when the MIS is up and running—nothing could be further from the truth. The MIS, it must be remembered, is not equivalent to the goal.

By careful attention to the steps in the Implementation Procedure Diagram, you can make your MIS fall into the 30% success group. But management must be committed to its presentation and use. During the conceptualization activities, a manager should consider the desirable extent and sophistication of the sys-

tem. This will ensure resources and commitment are sufficient to complete all the steps in the Implementation Procedure Diagram. And, of course, flexibility is also a key—a good smaller system is a better result than a sophisticated system failure.

The nine steps in the Implementation Procedure Diagram are needed for successful MIS development, but their completion does not guarantee a desired result. Since a manager must depend on MIS to give personal and crucial service, a comparable measure of energies must be given to the development and implementation of the system.



---

## 2.1.4. CLASSICAL APPROACHES AND THEIR HYBRID DESCENT.

---

**Use a dual path approach to implement the system development life cycle so you can make the most out of the advantages of both classical approaches.**

Figures 1.1.20.1.3., 2.1.3., and 1.1.20.1.1.a. and 1.1.20.1.1.b. represent a hybrid of the two classical approaches to system development. Because of Figure 2.1.3.'s attention to conceptualization, emphasis on prototype development, and detailed implementation stage, it offers a hybrid approach to ensure the MIS meets the manager's information needs.

There are two classical approaches to system development. These are the evolutionary (or bottom-up) approach and the systems (or top-down) approach. The bottom-up approach came first and was primarily responsible for operational automation. The top-down approach came into the MIS world when computers began to be of use to higher levels of management.

The bottom-up approach states that the way to develop an overall system is to start with lower-level functions (such as file updating and transaction processing), and progress to more conceptual considerations (such as control and decision modules).

The advantages of the bottom-up approach are (1) it proceeds step-by-step, in accordance with demand; (2) it allows for early gratification with a preliminary product at little cost; (3) it builds on transaction processing; (4) it minimizes the risk of building a large-scale system which doesn't operate properly; (5) the overall probability of failure is reduced, because smaller, simpler systems are being worked with; and (6) there is less likelihood of developing an overly sophisticated system.

However, the bottom-up approach has as many disadvantages as it does advantages. The main disadvantages are (1) as management's needs change, system redesigns are frequent; (2) if one section of management decides not to be integrated into the system, the evolutionary process is halted; (3) data inconsistencies are more probable, once the system becomes more comprehensive; (4) priorities are not easy to define; (5) the suborganization needs of the final system will be loosely integrated; and (6) the hardware supporting the various functions may not integrate easily.

The top-down approach states that the way to develop an overall system is to start with the organizational goals and objectives (such as improve personnel assignment and increase sales by 20%) and progress to more practical considerations (such as payroll and sales projections). The top-down approach assumes the systems needed to provide information can be developed once the information needs of management are determined. The approach seeks to develop a model of information flow in the organization and to design the information system to suit this information flow.

The advantages of the top-down approach are (1) there is allowance for greater development flexibility; (2) higher management is quickly committed and involved; (3) the total organization is included in the design; (4) planning is done strategically, rather than on a strictly operational basis; (5) new data are easily integrated into the system, again because of the flexibility of the design; and (6) all the ele-

ments of the management system are considered.

There are a number of disadvantages to developing a management information system in this way; notably, (1) it's difficult to derive the system plans from the objectives and activities of the organization; (2) it's difficult to assign cost and value to the modules; (3) the order of module development isn't necessarily related to organizational support or most potential use; (4) management loses faith when they don't see early results; and (5) there's a terrifying risk of building an enormous system which doesn't work properly.

When evaluating the pros and cons of the two classical approaches, the tool builder and user must consider several considerations. You must balance risk and results. Will the results be small, periodic, and early? How clearly and correctly do you understand what the final system should do or will intermediate products provide helpful feedback? How tangible are the results, whether early or large, infrequent, and late? What's the risk of improper fit in terms of size, sophistication, and applicability of the system? What's the advocacy for the system? High-level? Fickle or solid? What are the needs for system integrity and integration? Is the first focus one of functional tasks or one of organizational objectives? How will the system evolve?

From the classical approaches, we've been

and will continue to develop hybrids. The idea, of course, is to employ the best contributions from the two classical approaches. At Management Systems Laboratories, we call the hybrid approach a dual-path approach, to indicate that both bottom-up and top-down principles and activities are used to enhance the advantages, and neutralize the disadvantages, of each.

Figure 2.1.3. emphasizes the strict development of organization-wide system specifications through Situation Analysis, thus ensuring that a designed system will meet the needs and desires of all levels of management, from operational to strategic. At the same time, Figure 2.1.3. provides for early system prototypes, thus allowing the immediate gratification provided by the bottom-up approach.

Basically, the steps of Figure 2.1.3. allow the manager to plan from the top-down, and design from the bottom-up. Several smaller functions can be designed while the complete system is being planned.

We haven't been able to use a dual-path approach for too many years. Ten years ago, the cost of programming and databases would have made it prohibitive. The modern relational database structures and other technical developments have relieved the constraints that kept the dual-path approach from being cost-effective.



---

## **2.0. BUILDING MANAGEMENT TOOLS**

### **2.1. APPROACHES FOR BUILDING TOOLS**

#### **2.1.5. ANALYZING INFORMATION FLOW**

---

### 2.1.5.1. DIAGRAMMING INFORMATION FLOW.

---

**By diagramming the flow of information between conversion processes, we can get to the data elements we need about our work process.**

#### **Charting**

Charting is a tool we use to help gather information, make sure it's complete, verify its accuracy, and communicate the ideas we've charted. We're familiar with many types of pictorial representations. As engineers, we draw parts we want machined. As managers, we chart our organization to see who reports to whom. Remember the pictorial representation, or chart, is only as good as the data it's based on. The chart can quickly get out of date, and the chart can be drawn from bad information (or data) in the first place.

#### **Flow Diagramming**

When we chart, or diagram, the flow of something, we're interested in identifying the steps in, or the flow of, a process and in showing precedence. We want to know what comes before (predecessor) and what comes after (successor). Since charts are two dimensional (or, at best, three dimensional), it's hard to completely capture a multidimensional situation like management. We pick what we want to show and develop a convention for how we chart things, so we can show what we want to show consistently. As long as we're consistent, any convention will do. By convention, I mean what symbols we use, how we number or name things, how various charts link together, and so on. How consistent we have to be depends on what we're going to do with the chart. Must the entire organization use the same convention in everyone's charting? How about everyone in the discipline (e.g., industrial engineering)?

In flow diagramming, we show something flowing into or out of a process. We can worry first about what flows (in our case, informa-

tion) and later about what the flow goes into and out of or what happens to the flow inside the thing (in our case, information conversion processes) it goes into. Or we can worry first about the things that transform the flows (what they are and what they do) and later about what the flows are that are affected by the transformations. Whether we start with the flows or the transformations is a matter of preference; and, in some organizations, may be a convention. In information flow diagramming, some books discuss only one preference and the other books discuss only the other preference.

Flow diagrams also show sources (where flow comes into the picture from) and sinks (where the flow goes out of the picture to). (Remember, the terms sources and sinks come from heat transfer. Sources are where heat comes from. Sinks are where heat goes.) In addition, flow diagrams show accumulations (merging) of the flows and distributions (extracting) of the flows. Flow diagrams aren't logic diagrams. They aren't supposed to show branching, or choice. A flow diagram may show several pieces of information or multiple copies of the information coming into or out of a conversion process. The flow diagram won't show a decision determining what information is sent and won't convey the idea of following one path over the other based on the decision.

#### **How to Use Flow Diagrams**

Flow diagrams will help you improve your gathering information about the organization (or whatever) you're studying. You'll gather all kinds of bits and pieces of information. True understanding of what you're studying depends on your ability to put the bits and

pieces together, and to do so correctly and completely.

A flow diagram helps you sort out the bits and pieces and to try to relate them to each other in different ways. You should find out what the flows or conversion processes are from the people who do the flows and processes and know them best. Then link them together by showing flows as arrows and conversion processes as circles or ovals. Now you have an integrated picture. Check to see if the person Fred said got a piece of information from him in turn said they got that information from Fred. Can you see what's going on? If not, you're probably missing something—or it's wrong. Is the information flow easy to understand? Does a portion of the flow not make sense to you? Now look at what you think is going on and see if you have any questions; then go ask more about what you're not sure of.

When you verify if you captured the ideas correctly or ask for more information, use the flow diagram. Show the diagram to the people you gathered the information from and see it makes sense to them. You may have problems with confusion between information about what the flows are or about what the people you're getting the information from would like the flows to be. You also may have a hard time getting several people to agree on any one version you show them. You'll have to decide when you know what's going on "well enough," and, at the right time, abandon your iterations in improving the flow diagram. Knowing when to abandon your iterations comes from experience.

### **Information Flow**

Consider a domain of responsibility. Delimit that domain. Now draw an oval representing the boundary of the domain. Show information flows into and out of the domain by drawing arrows across the boundary. All information flows are shown as arrows. The

arrows are like pipelines, and the information can flow back and forth. Many information flows are not one-directional. That is, in the transfer of the information, there is give-and-take between the sender and receiver. Usually, however, the flow is predominantly in one direction. You can show the dominant direction with an arrow head.

Let's focus on information-flow-type diagrams and develop some convention. Many books like to use the term data flow diagrams (DFD). I prefer information flow diagrams because information is what's transferred. Information, however, is made from data. The information portrayal format (the thing that moves, or flows) contains selected data elements.

Figure 2.1.5.1.a. shows the symbols we'll adopt as convention. The arrows are the information flows and the circles, or bubbles, are the information conversion processes, or the processing functions. These diagrams are sometimes called bubble diagrams. Sometimes a flow diagram can get quite large or have a lot of flows. To keep from getting too much on a page or from having a lot of crossed lines, we'll use small circles as transfer indicators. If an arrow goes to a small circle with a four in it, look for another small circle with a four in it and an arrow going away from the circle. These circles pictorially link the same information flow.

We show an organization *outside* our domain of responsibility that sends information to or receives information from our domain as a square. We often store information (and data). A three sided rectangle represents a file, or data store. Finally, we use the triangle to show merging and extracting information flows.

### **The Context Diagram**

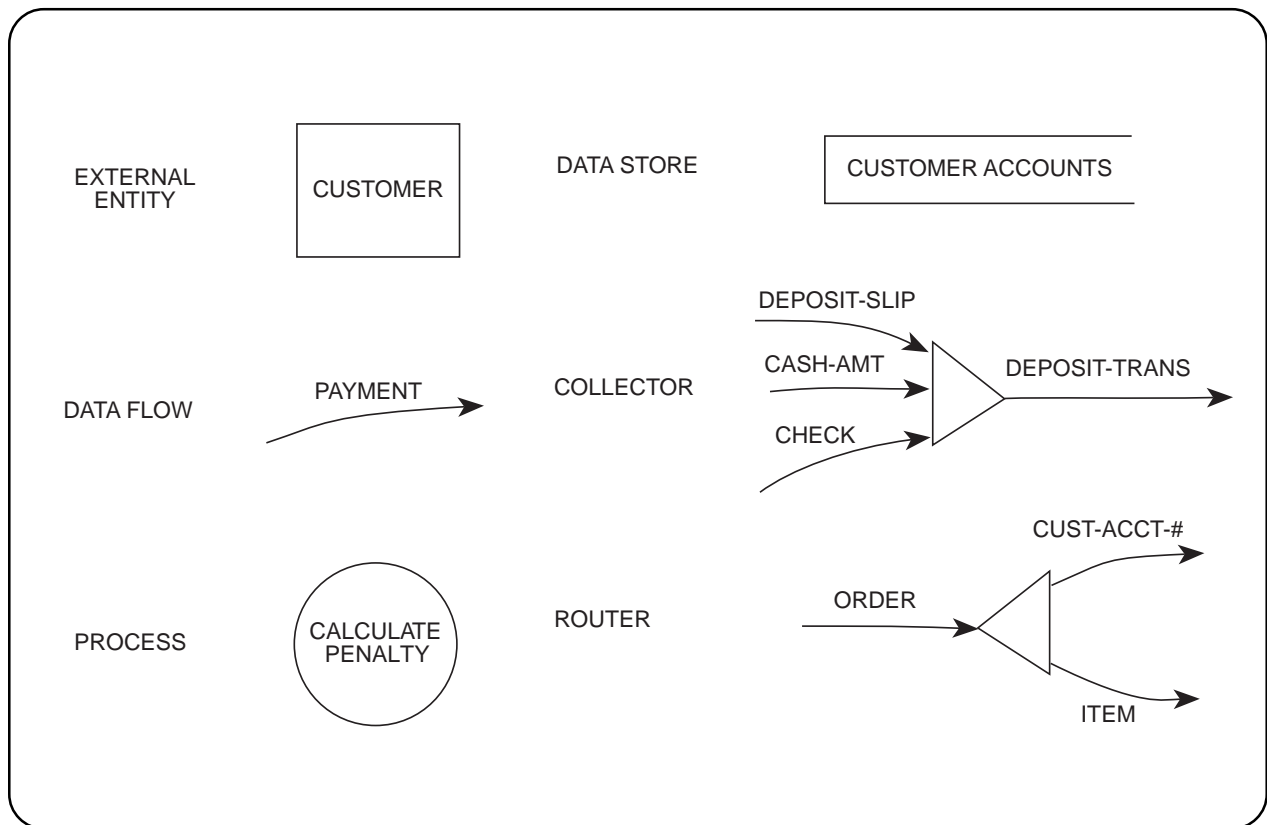
In Figure 2.1.5.1.b., I've shown the domain of responsibility for a human resource management department in the engineering division of a company. I've shown the domain as an

oval and included only a few simple examples of the context of this domain. That is, I've shown just a few of the information flows into and out of the department and the external entities related to those flows.

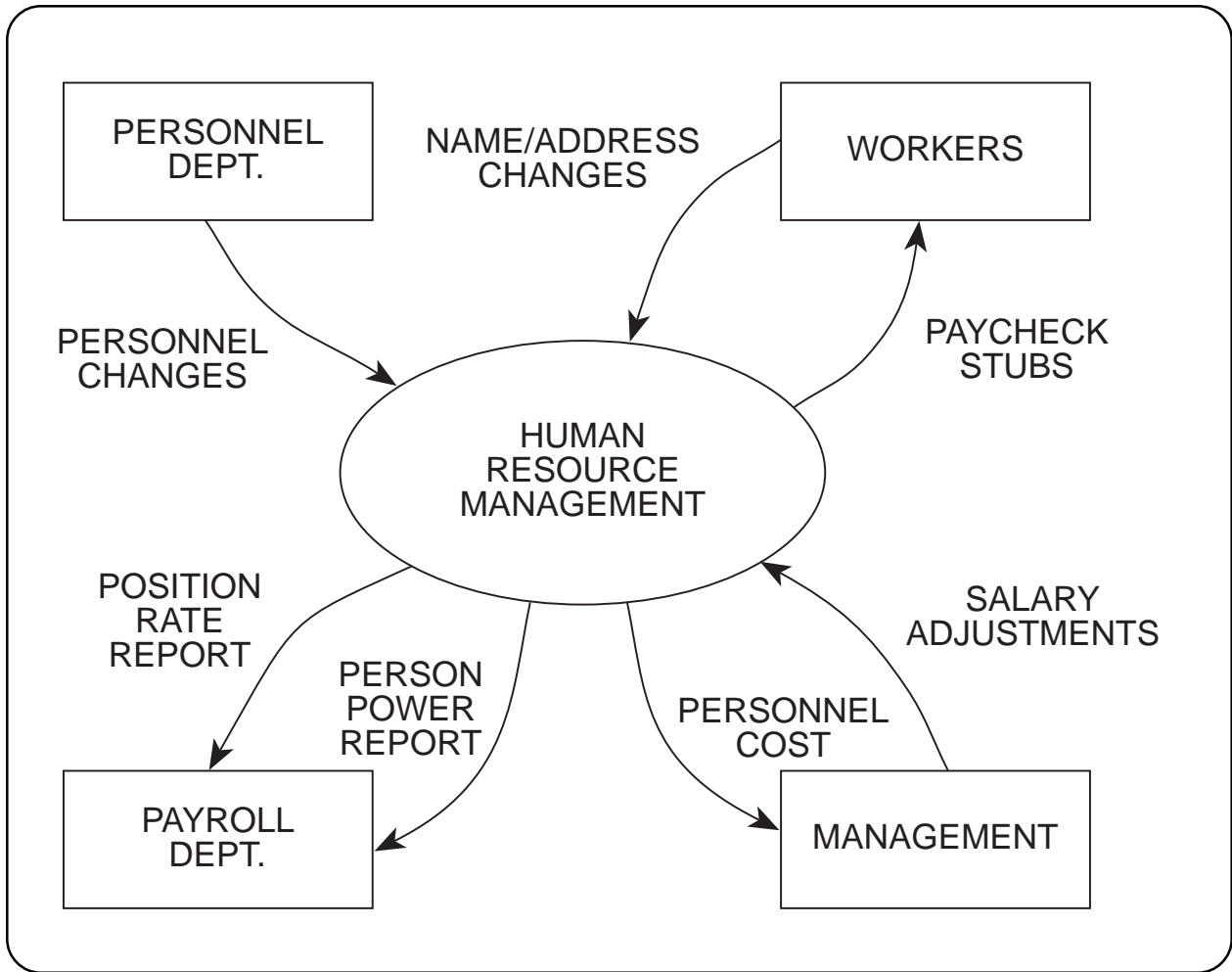
The diagram in Figure 2.1.5.1.b. is called a context diagram because it places the domain in context with its environment. Notice that the diagram doesn't include arrows between any two of the external entities—because we're not concerned with information passed between them. These information flows aren't in our domain of responsibility. Sometimes, we don't show what the information flows are in a context diagram. (We don't label the arrows.) Then, we're usually trying to delimit our domain. Sometimes we do label the ar-

rows. Then, we're using the context diagram as a first step in information flow diagramming.

Also take note that all arrows either *come from* an external entity or *go to* one; there are no arrows going away from the system into thin air or coming from thin air. This representation is accurate because, in real life, all information sent out of a system has to go somewhere—it doesn't vanish. And conversely, information coming into the system has to come from somewhere—it doesn't just appear on your doorstep. Some authors don't include boxes for some external entities because they feel some entities are so well known, they're obvious. For clarity, I prefer to include all external entities you can think of.



**Figure 2.1.5.1.a.** Data flow diagram symbols and meanings. (adapted from Powers, Adams, & Mills, p. 258)



**Figure 2.1.5.1.b.** An example context diagram shows the domain in relation to its environment.

## 2.1.5.2. DIFFERENT LEVELS OF DATA FLOW DIAGRAMS.

---

**By continually partitioning a conversion process, you can develop information flows to the detail you need to identify all data elements.**

### Partitioning

After you've created a context diagram for the system, your next step is to think about the information flows *within* the system. But since there are so many flows inside the domain, you must break down, or *partition*, the domain into subdomains so you can track the information flow in greater detail. This partitioning involves creating an information flow diagram for each of the subdomains. I'll call these level-1 diagrams. You'll continue partitioning each process, and thereby create new sublevels, until you have a suitable representation of all the information flows. Each of the subdomains will bear the number of the higher level from which it came plus an additional digit for the current level. (See Figure 2.1.5.2.a.) Just remember: processes have 1 digit at level-1, 2 at level-2, and so on.

To sum up the system levels, understand that there is only one context diagram, one level-1 diagram (processes numbered like 1, 2...), several level-2 diagrams (1.1, 1.2...), and many more level-3 diagrams (1.1.1, 1.1.2...), for however many levels you need to capture all information flows. See Figures 2.1.5.1.b., 2.1.5.2.b., and 2.1.5.2.c. to help you visualize the levels of diagrams .

Figures 2.1.5.1.b., 2.1.5.2.b., and 2.1.5.2.c. all refer to the same domain. Figure 2.1.5.1.b. is the context diagram. Figure 2.1.5.2.b. is the level-1 diagram showing the human resource management department partitioned into four major parts. This division was made by function. We can divide, or partition, the domain many different ways. The important thing here is all the subdomains must total the entire domain.

One common way to partition is by functions. Other ways include partitioning by process steps, by customer types, by vendors, by product, and by geography.

Figure 2.1.5.2.c. is the level-2 diagram (two digits identify the sub-subdomains). Figure 2.1.5.2.c. is the domain of the person responsible for computing pay. The number 2 and 4 domains are outside the domain of computing pay but part of the human resource management domain. In Figure 2.1.5.2.c., these domains are shown as external entities and identified by the numbers of the conversion processes in Figure 2.1.5.2.b.

We can adopt the convention that entities external to the subdomain but internal to the domain are identified by their number. If one of the information flows from outside the larger domain had flowed into the computer pay process (Notice from Figure 2.1.5.2.b., none do.), we would show that in Figure 2.1.5.2.c. as a square with the name of the external entity written in it.

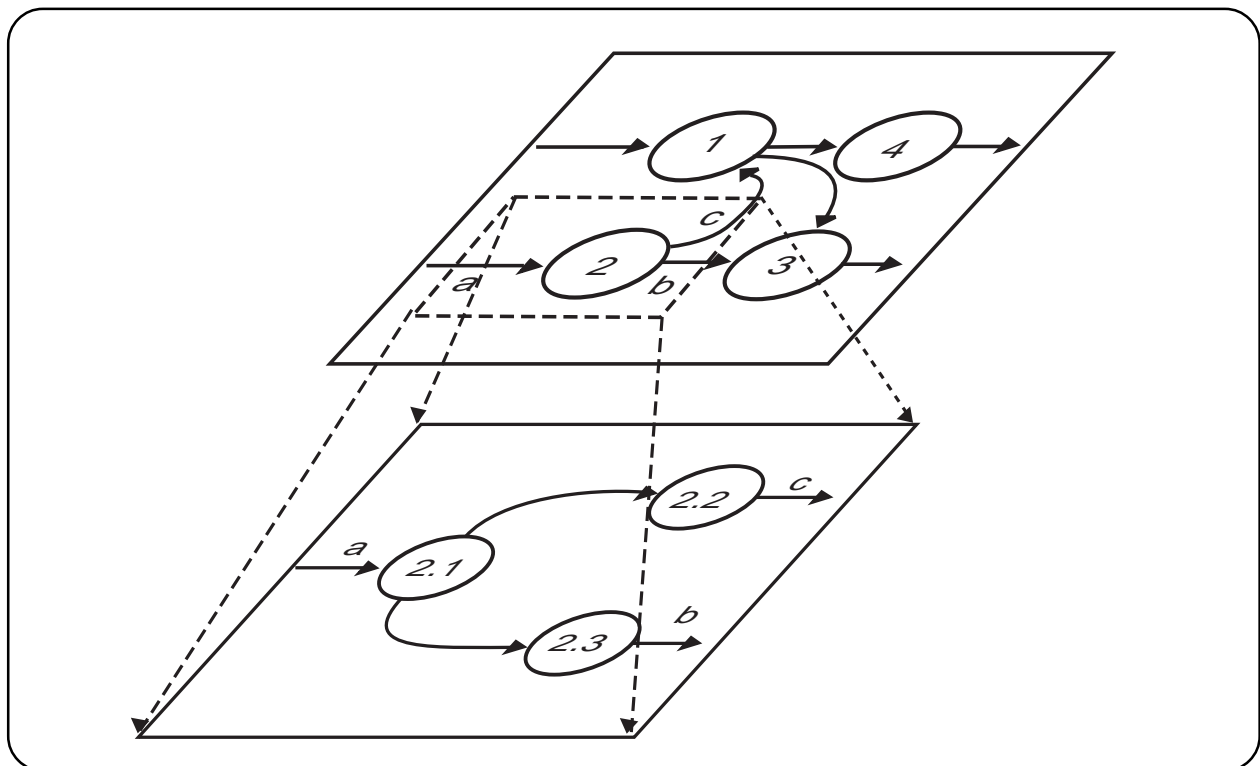
I'll recap our partitioning discussion. We begin with our entire domain of responsibility as a context diagram. Remember that since the domain is a management *system*, it's already part of a larger system. We divide, or partition, the domain into subdomains. We can partition any number of ways. We choose the way that helps us the most. I chose the way the book I took the figure from did it. We partition the domain into subdomains. If our original domain is large enough and we partition the domain like the organization chart is structured, each subdomain is someone's domain of responsibility.

We can further partition each subdomain into logical parts—all adding up to the whole. And we can partition any or all of those parts, or subdomains, into their parts, or sub-subdomains. We continue doing this until we've identified every information flow and information conversion process in the original domain. We'll end up with many information flows. And each information flow has many data elements in it. *It's the data elements that the MIS developer wants to identify so he or she can figure out ways to acquire, store, retrieve, and manipulate those data.*

When you try to do a data flow diagram (DFD) or information flow diagram and start with a blank sheet of paper, what do you do? One book (Yourdon) says you should draw in the information flows and then look at the flows going into or out of a process to figure out what the process is and to name it. For example, in Figure 2.1.5.2.b., if a valid transaction flows in

and a sorted transaction flows out, the conversion process is the “sort transaction” process. Another book (Powers, Adams, and Mills) suggests starting with the process bubbles and then figure out what the flows are. I've tried both and usually end up doing it the Powers, Adams, and Mills way. Maybe what a person chooses to do depends on personality type.

To give you another look at partitioned DFDs, review Figures 2.1.5.2.d., 2.1.5.2.e., 2.1.5.2.f., and 2.1.5.2.g.. Figure 2.1.5.2.d. is the context diagram for the domain of responsibility called APT (Astro-Pony Toutshops) which is, from what I can tell, a bookie. Figure 2.1.5.2.e. is the level-1 diagram for APT. APT also has been partitioned by function. You'll notice this looks more like the second level of APT's organization chart. Look at the external entities on the level-2 diagrams. Use the external entities to find inconsistencies and, therefore, potential problems.



**Figure 2.1.5.2.a.** We can partition domains into subdomains, subdomains into sub-subdomains, etc. (adapted from deMarco, p.72)

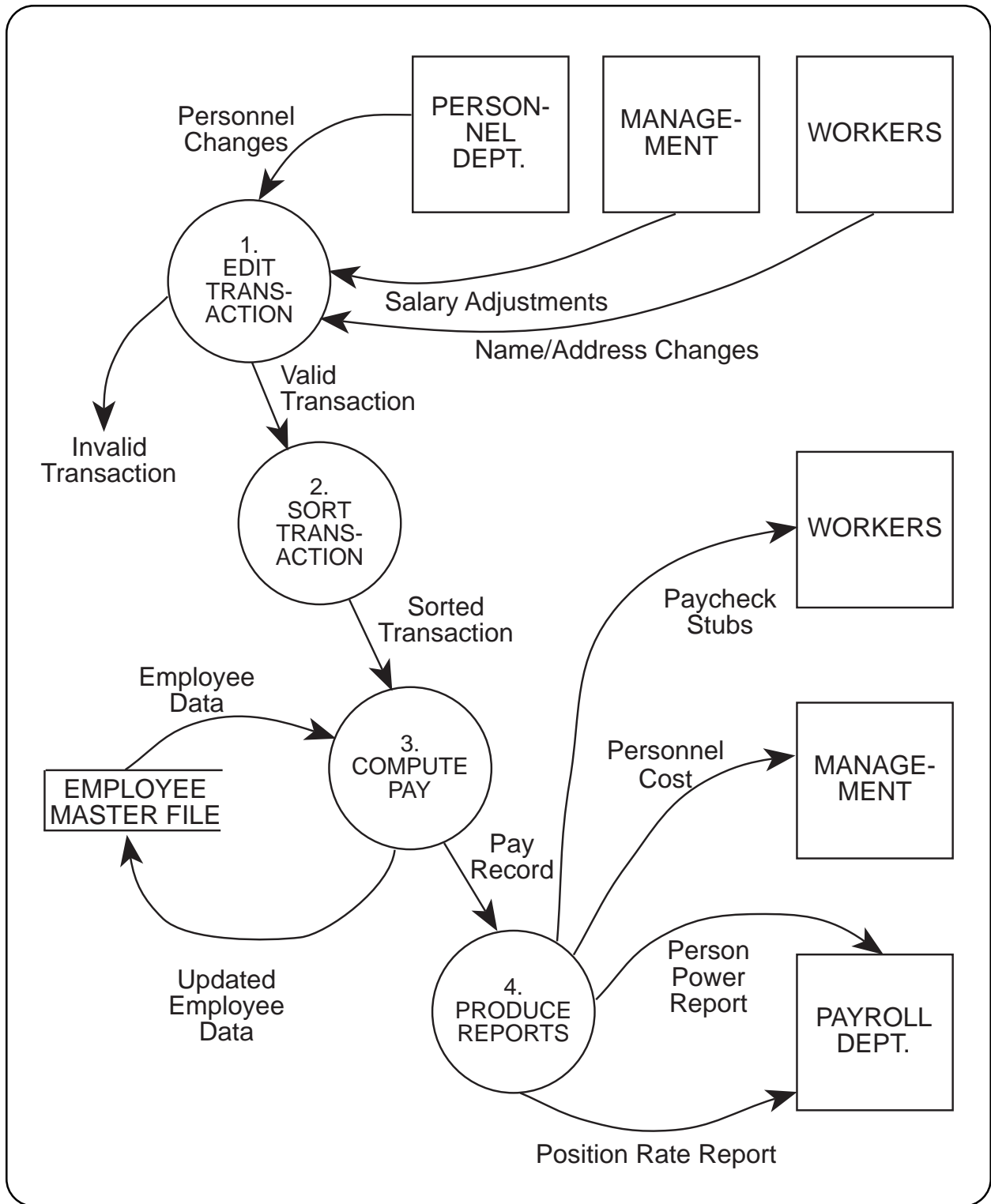
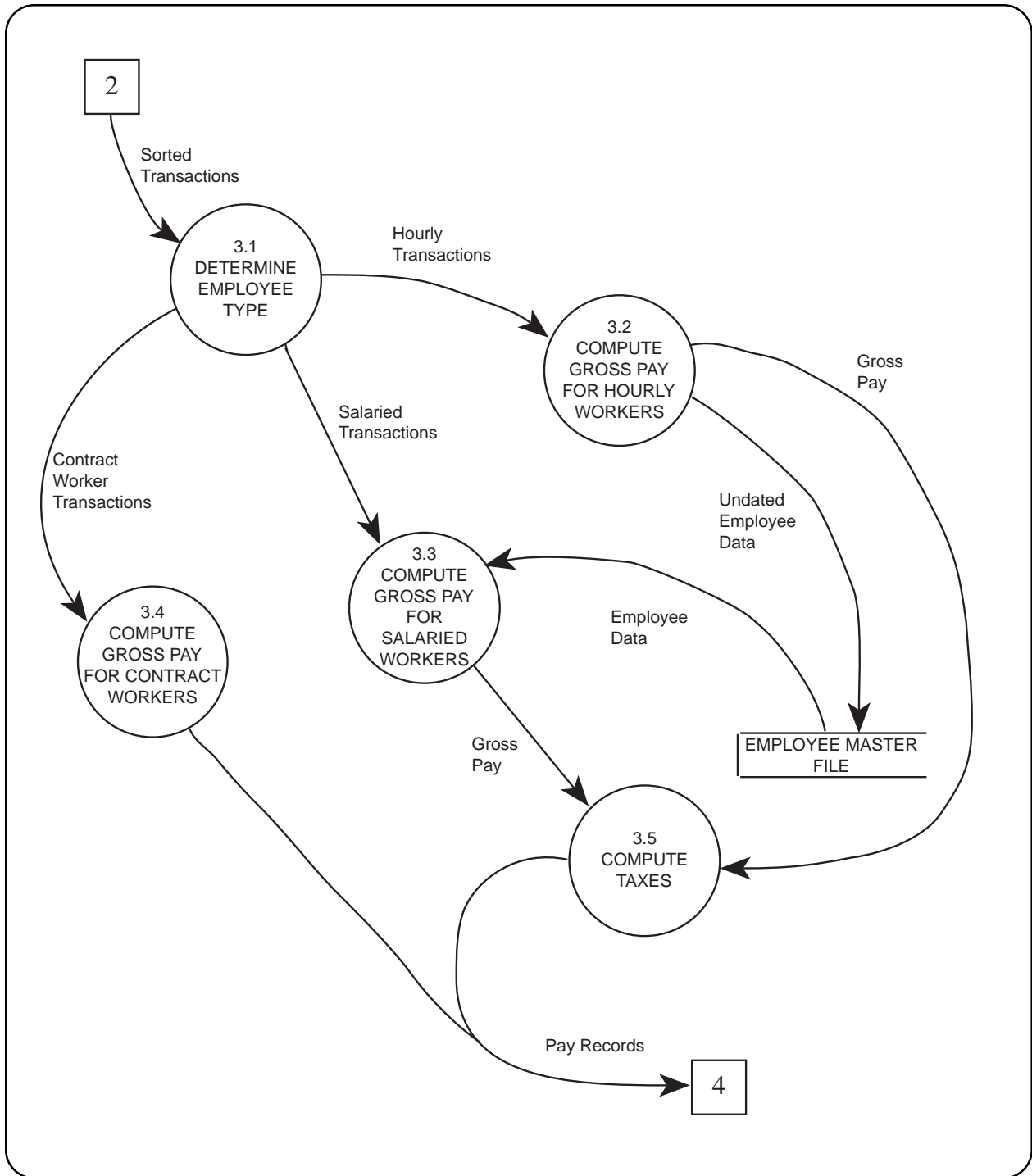
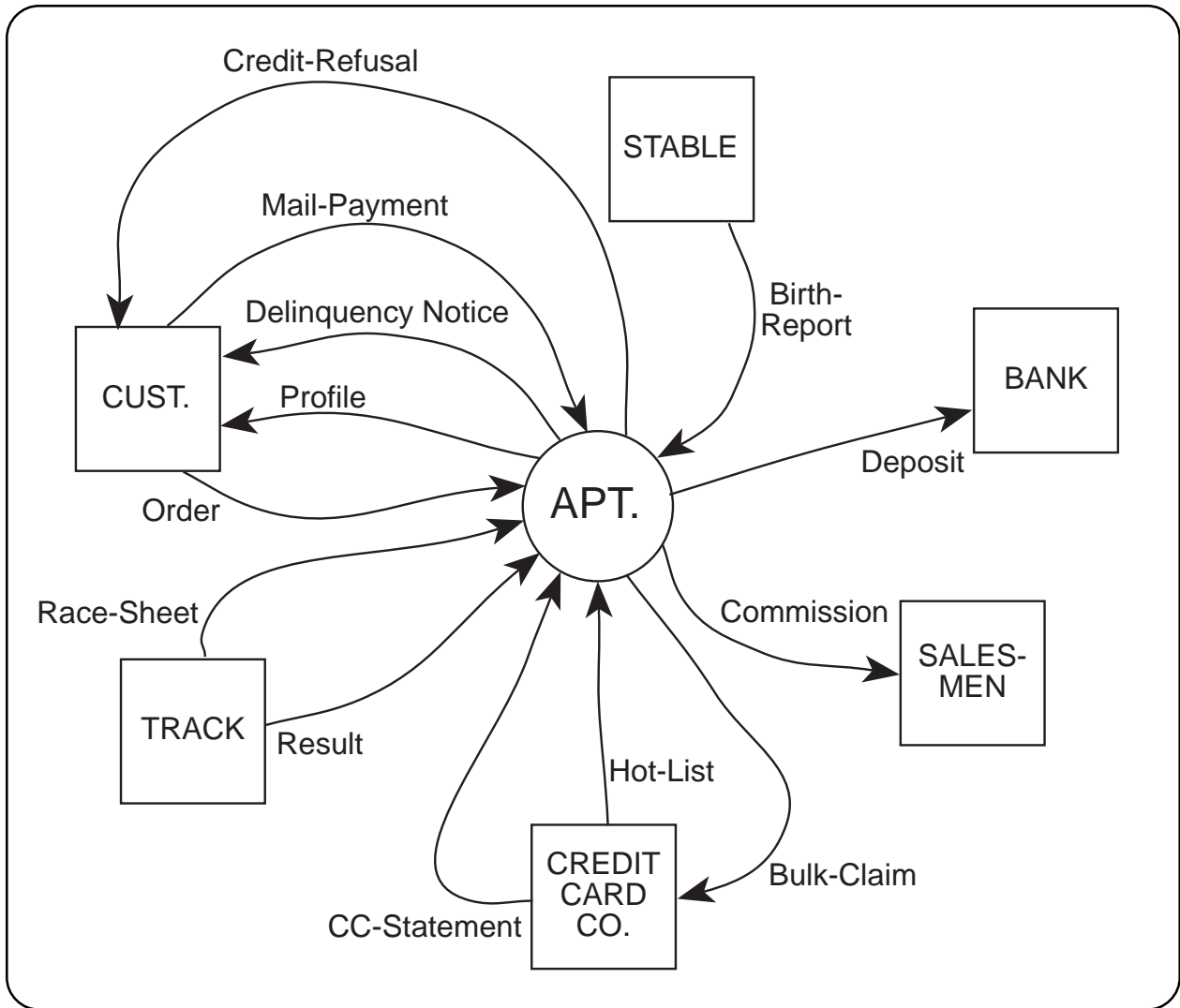


Figure 2.1.5.2.b. An overview data flow diagram. (adapted from Yourdon, p.13)



**Figure 2.1.5.2.c.** A detailed data flow diagram. (adapted from Yourdon, p.14)



**Figure 2.1.5.2.d.** Example context diagram. (adapted from deMarco. p.90)

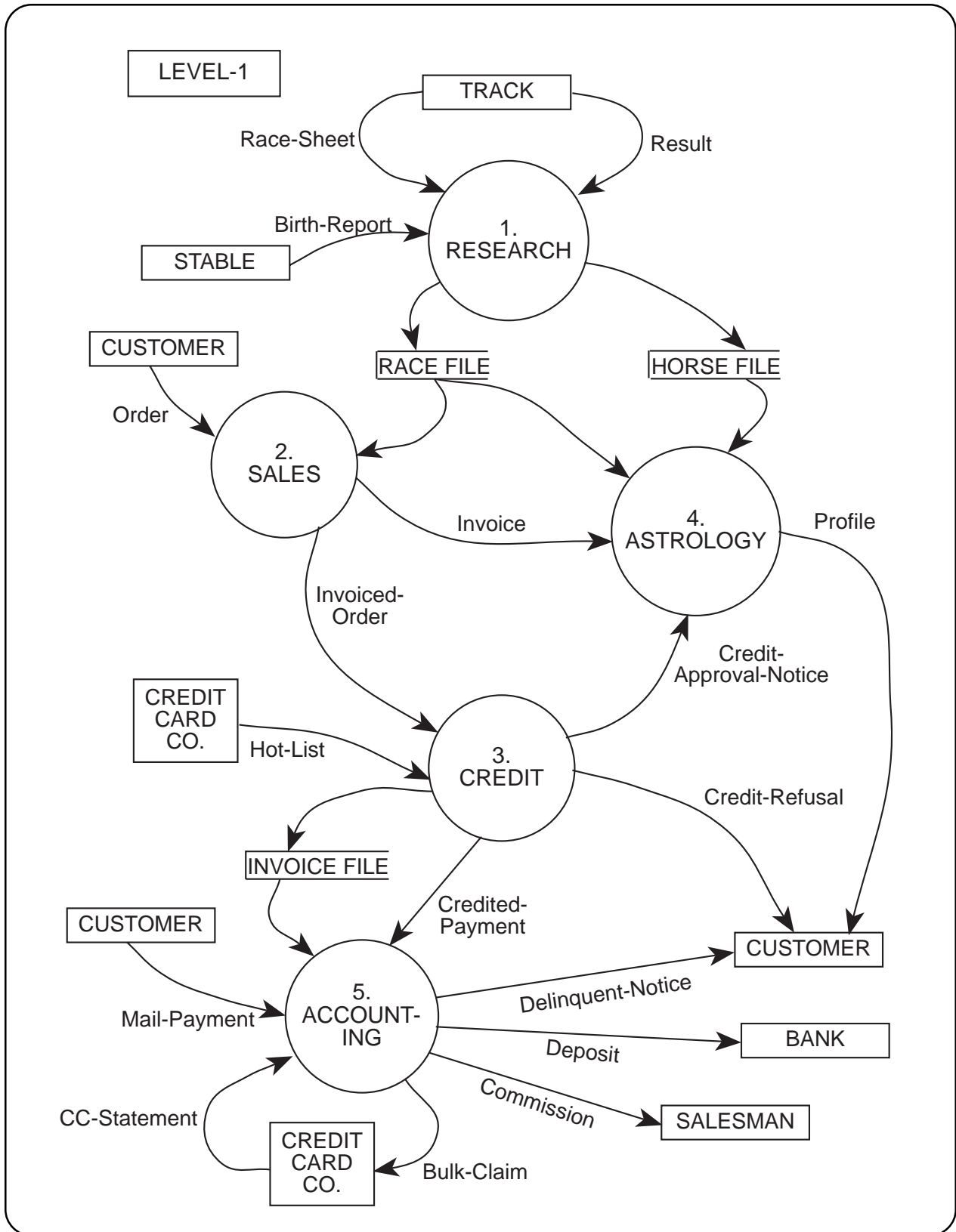
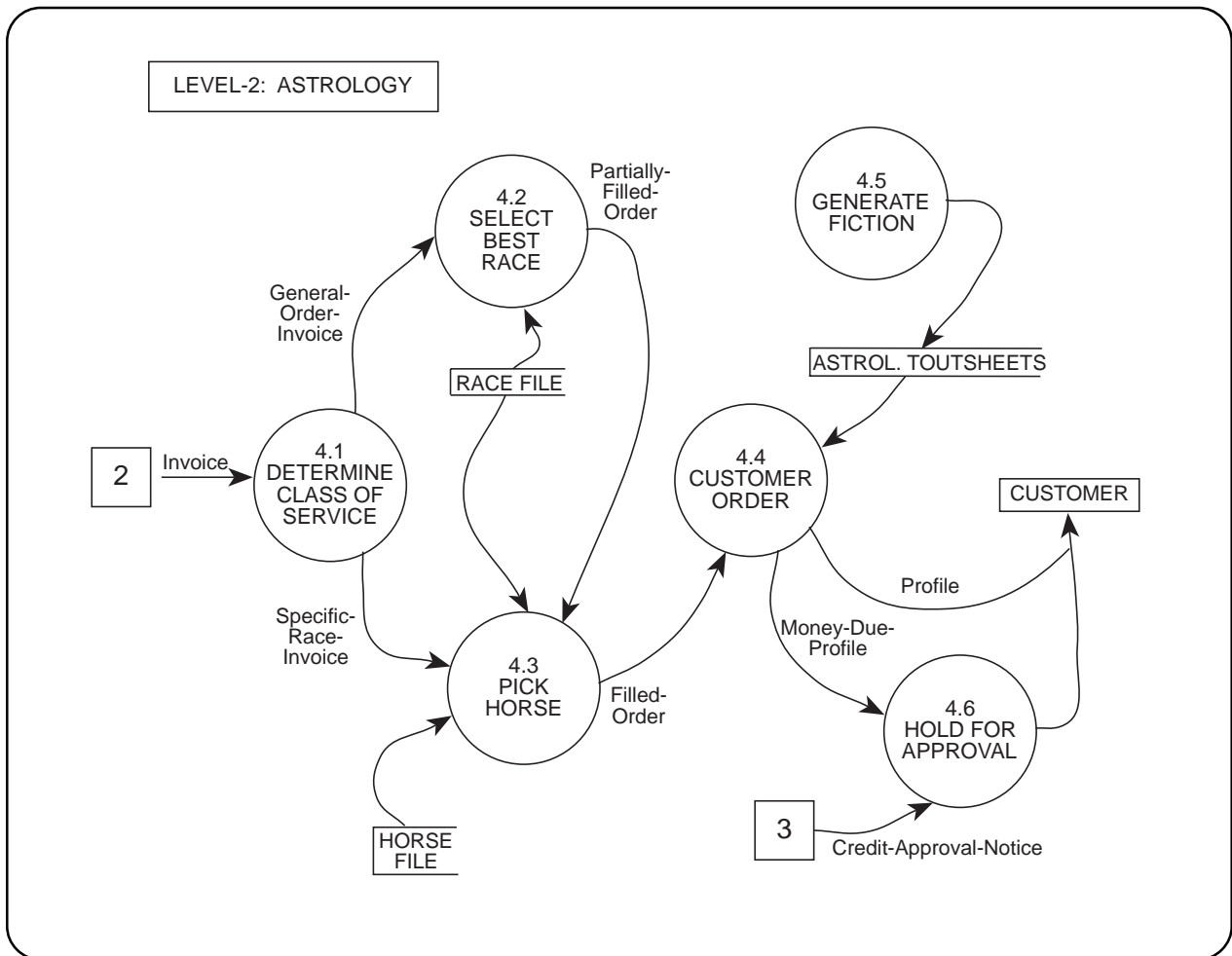
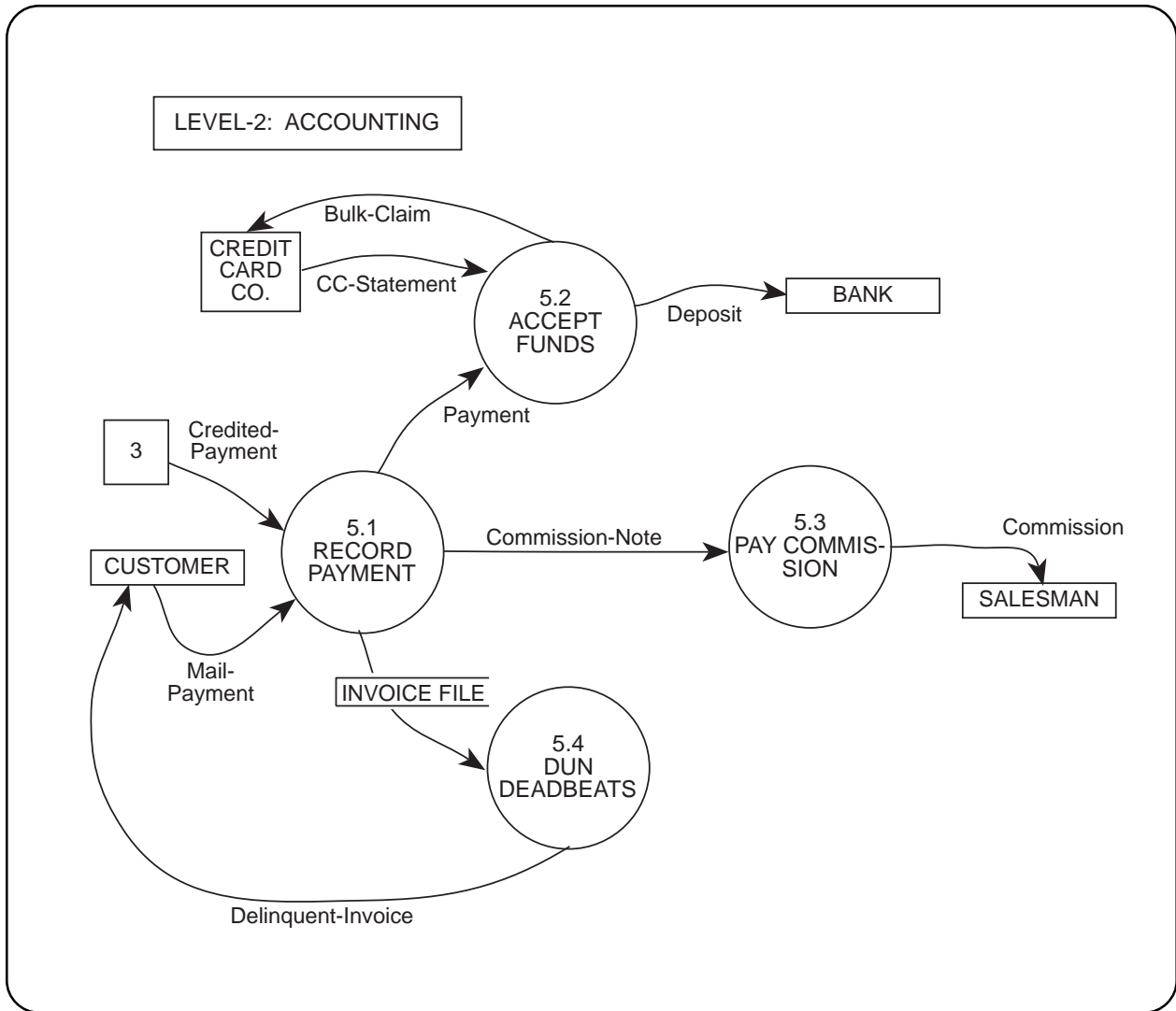


Figure 2.1.5.2.e. Level-1 Data Flow Diagram. (adapted from deMarco, p. 92)



**Figure 2.1.5.2.f.** Level-2 Data Flow Diagram. (adapted from deMarco, p.97)



**Figure 2.1.5.2.g.** Level-2 Data Flow Diagram. (adapted from deMarco, p.101)

### 2.1.5.3. PHYSICAL VERSUS LOGICAL DIAGRAMS

---

**Physical information flow diagrams are best for gathering and verifying information; and logical information flow diagrams are best for communicating with automation specialists.**

The next major consideration in information flow diagramming is that there are two ways to chart information flow on an information flow diagram. These ways are physical and logical. Physical models focus on how a job gets done: the physical means such as documents, people and forms. Logical models represent what the system does and concentrate on the data and the underlying process used to manipulate the data. Look at Figures 2.1.5.3.a. and 2.1.5.3.b. Figure 2.1.5.3.a. shows a report flowing in a physical model. Figure 2.1.5.3.b. shows what happens to the same report in a logical model. Figure 2.1.5.3.c. summarizes the key differences between physical and logical models.

The physical model shows how the information progresses sequentially through a system. (See Figure 2.1.5.3.a.) Logical models, however, show that a piece of information can be acted on by more than one process at any one time—in a parallel fashion. (Refer to Figure 2.1.5.3.b.)

Physical models are good for making sure you've captured the information flows and the communication into, out of, and within a system. They're also easier to understand by a relative novice. He or she can look at the physical model and see department names or people and grasp what's going on in the domain of the diagram. However, sometimes on physical models, items that flow into or out of a process may not be logical—they actually occur, of course, but it's not always clear why. Logical models are better because they make clear the reasons why something flows in or out; they're good for automation and mechan-

ization of a process.

The logical model's drawback is that it is not readily understood by the average person. The symbols may be too complex.

Use the characteristics in Figure 2.1.5.3.c. to compare Figures 2.1.5.3.d. and 2.1.5.3.e. You can see an additional convention in Figure 2.1.5.3.e. The book from which I got the figure uses a single slash in the lower left-hand corner of an external entity to show that entity is repeated on the diagram. They repeat the student external entity because they don't want to draw such a long arrow from the external entity to wherever it's going. It's a matter of making the diagram look pretty.

Much of information flow diagramming is convention; that is, the symbols and other techniques vary from author to author. Each has his or her own preferences. I'm not saying here you should adopt my conventions; I just want you to be able to look at anyone's information flow diagram and understand it. Remember that a convention must be used *consistently* within a work to maintain clarity.

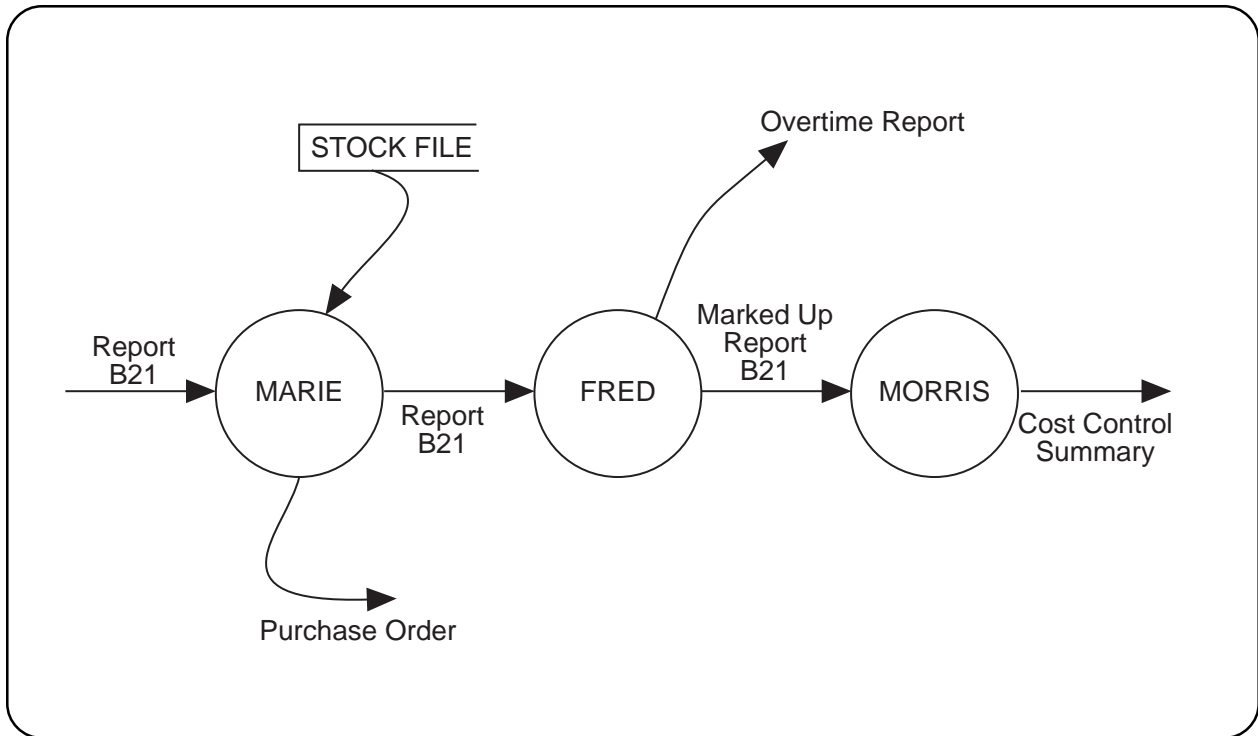
You can look at yet another example of physical and logical information flow diagrams. In Figure 2.1.5.3.f., you see a fairly cluttered physical DFD. This one is more typical. Notice how well you could use this diagram with the people you're gathering information from to see if you've captured what's going on. The symbols are simple. I advise drawing the external entities on the diagram, so you can verify them too.

In this case, you can see an intermediate step to converting physical DFD's to logical ones. Figure 2.1.5.3.g. is that intermediate step. The names have been dropped and the processes are more information conversion. Notice on Figure 2.1.5.3.f. that Jerry must have several duties and different kinds of documents come across his desk. Offices are like that. Often there's no rhyme or reason for the combination of things a person has to deal with. Figure 2.1.5.3.g. is still physical but a whole lot more logical.

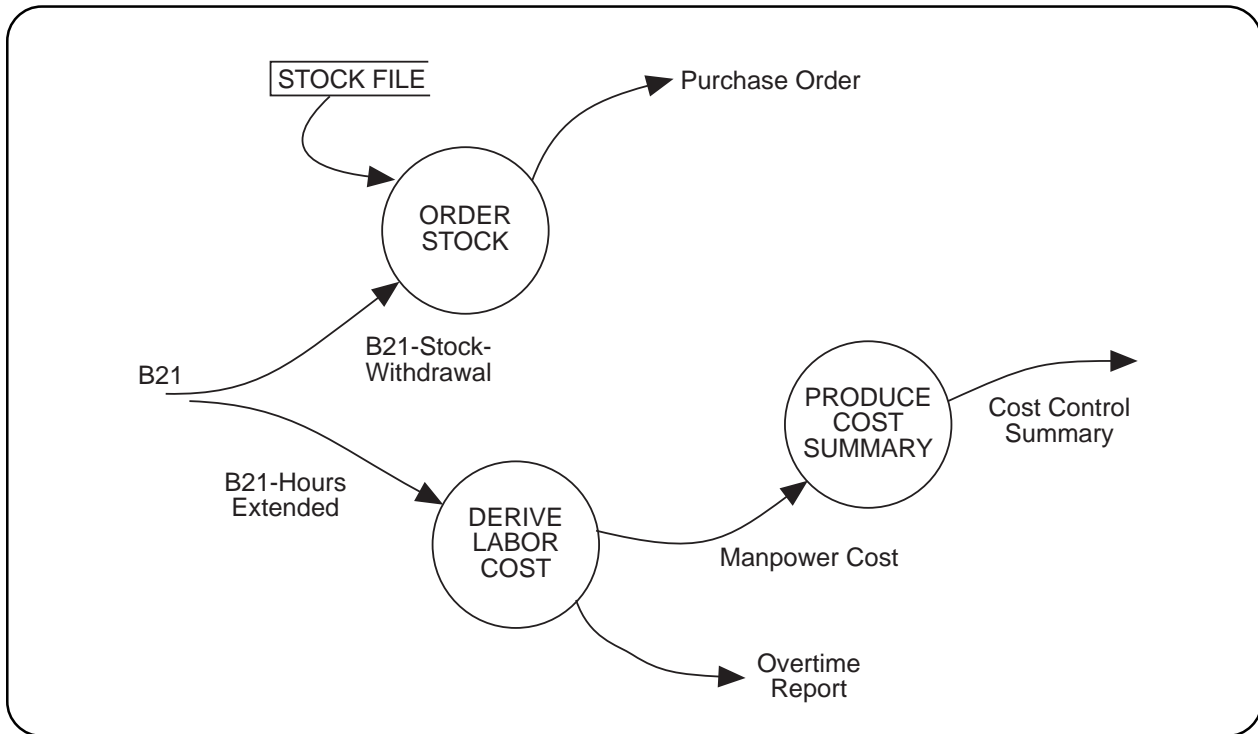
Figure 2.1.5.3.h. is the logicalized DFD. Now we're ready to talk to the computer programmers. We've analyzed the situation and put what we've learned into terms an automation

specialist can work with.

It takes some getting used to be able to capture the data, make physical DFD's, find errors and inconsistencies, review the information with the managers in the domain, convert to logical DFD's, and communicate with the automation specialists. For this book, I don't expect you to be able to do all this, but I do want you to be familiar with the concepts, the differences in convention, the differences between physical and logical diagrams, and the usefulness of the DFD's. I'll talk more about DFD's when we study system modeling. DFD's aren't the only tool we have to capture information about how an organization uses its information.



**Figure 2.1.5.3.a.** Example physical model. (adapted from deMarco, p.29)

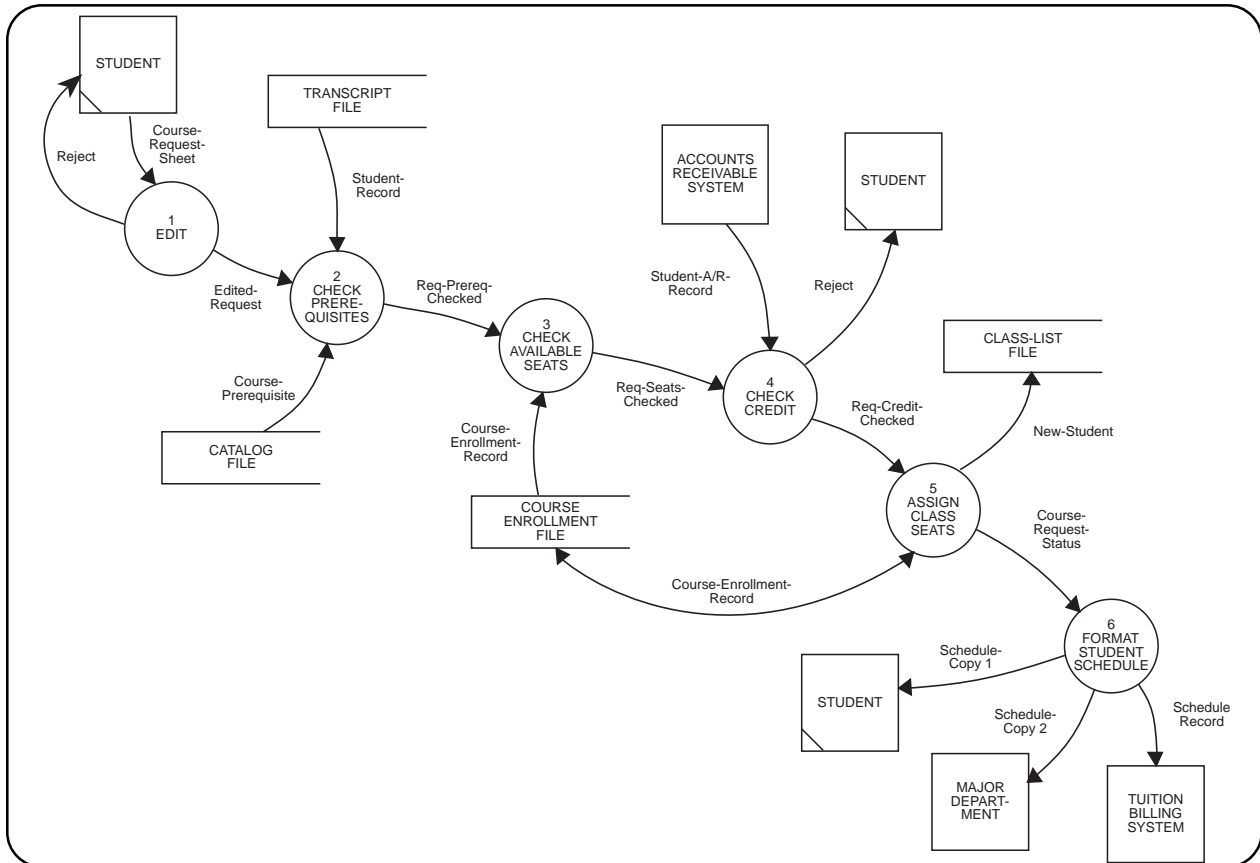


**Figure 2.1.5.3.b.** Example logical model. (adapted from deMarco, p.29)

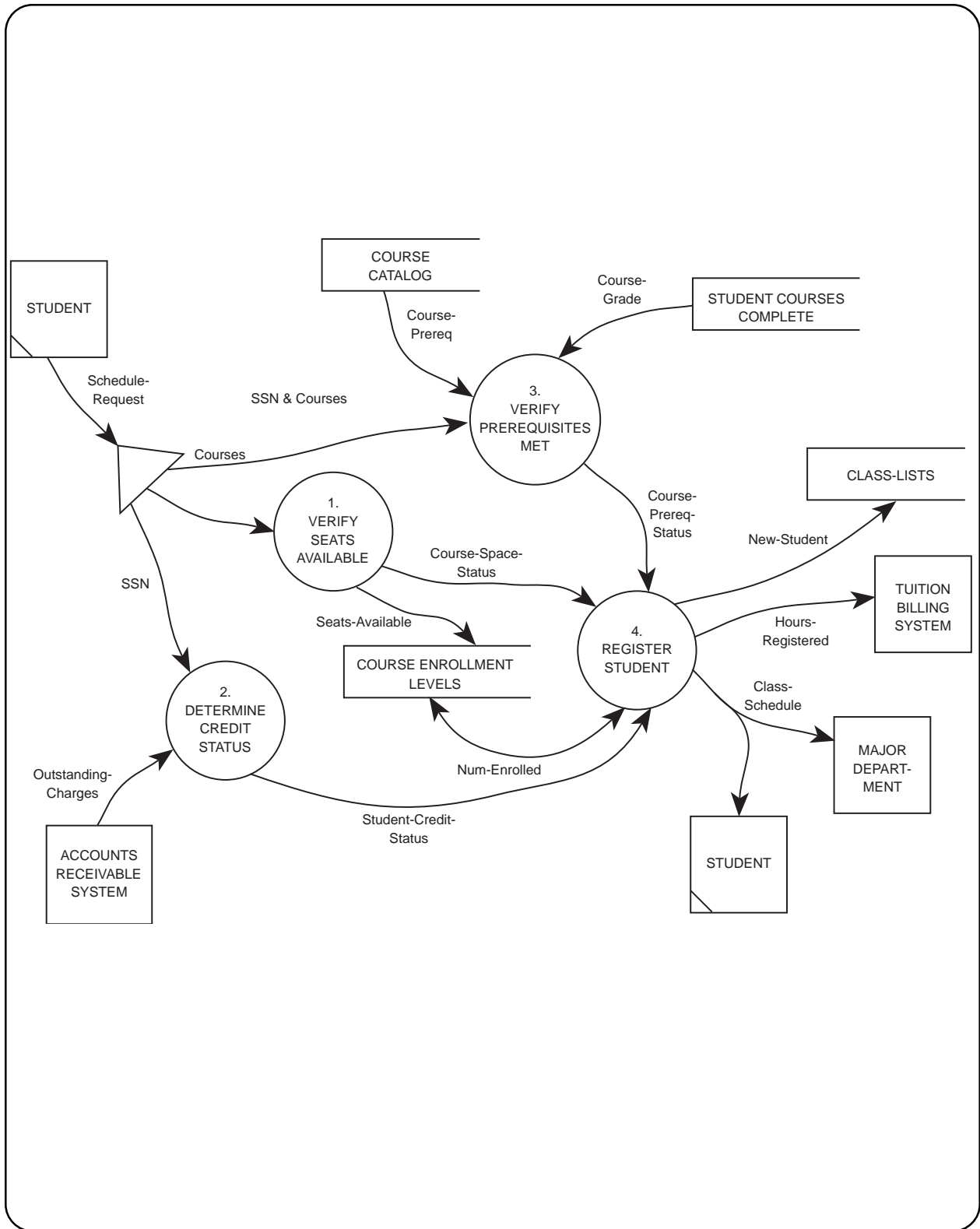
## MAJOR CHARACTERISTICS OF MODELS

	Physical	Logical
<b>Viewpoint</b>	How processing is done	What the system does
<b>Processes</b>	Sequential	Often parallel
<b>Names</b>	Documents, people, forms	Underlying data and processes
<b>Data Flows</b>	Excess (tramp) data	Only data used or produced by the process
<b>Controls</b>	Includes controls for crossing man-machine boundaries	Limited to essential business controls

**Figure 2.1.5.3.c.** Summary of key differences between physical and logical models. (adapted from Powers, Adams, & Mills, p.161)



**Figure 2.1.5.3.d.** Data flow diagram that emphasizes physical characteristics of a student registration system. (adapted from Powers, Adams, & Mills, p.162)



**Figure 2.1.5.3.e.** Data flow diagram that emphasizes logical characteristics of a student registration system. (adapted from Powers, Adams, & Mills, p. 163)

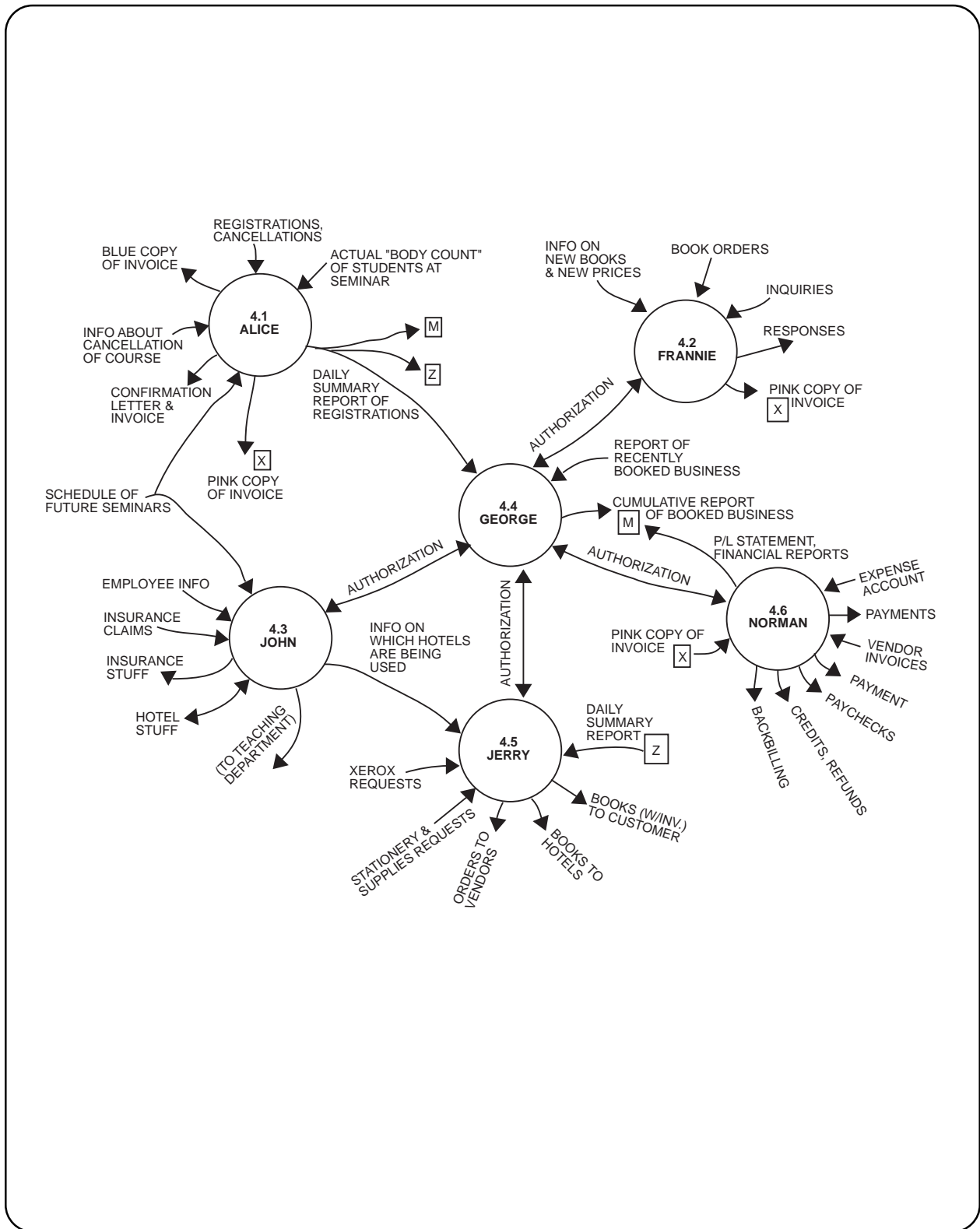


Figure 2.1.5.3.f. Current physical DFD. (adapted from Yourdon, p. 67)

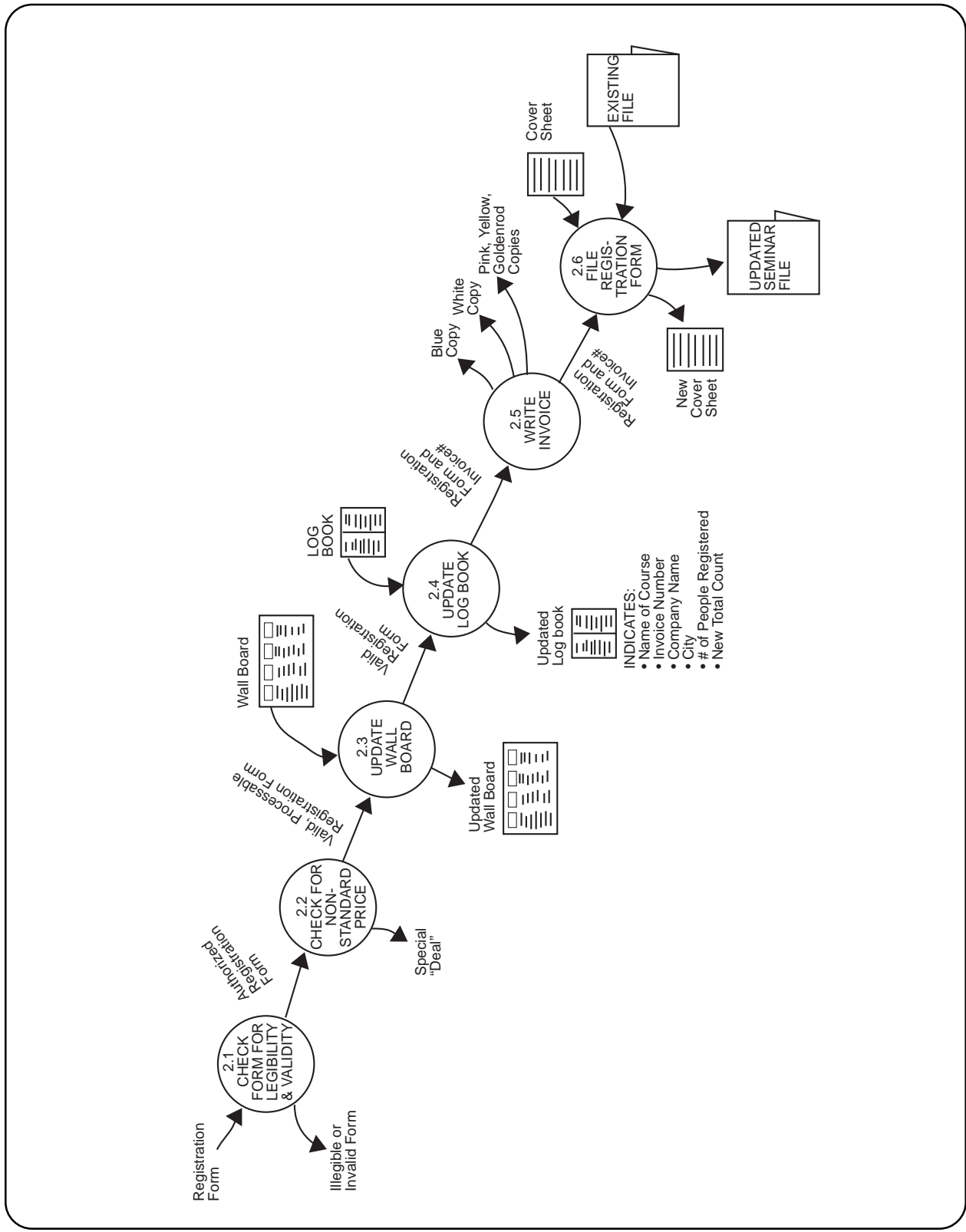
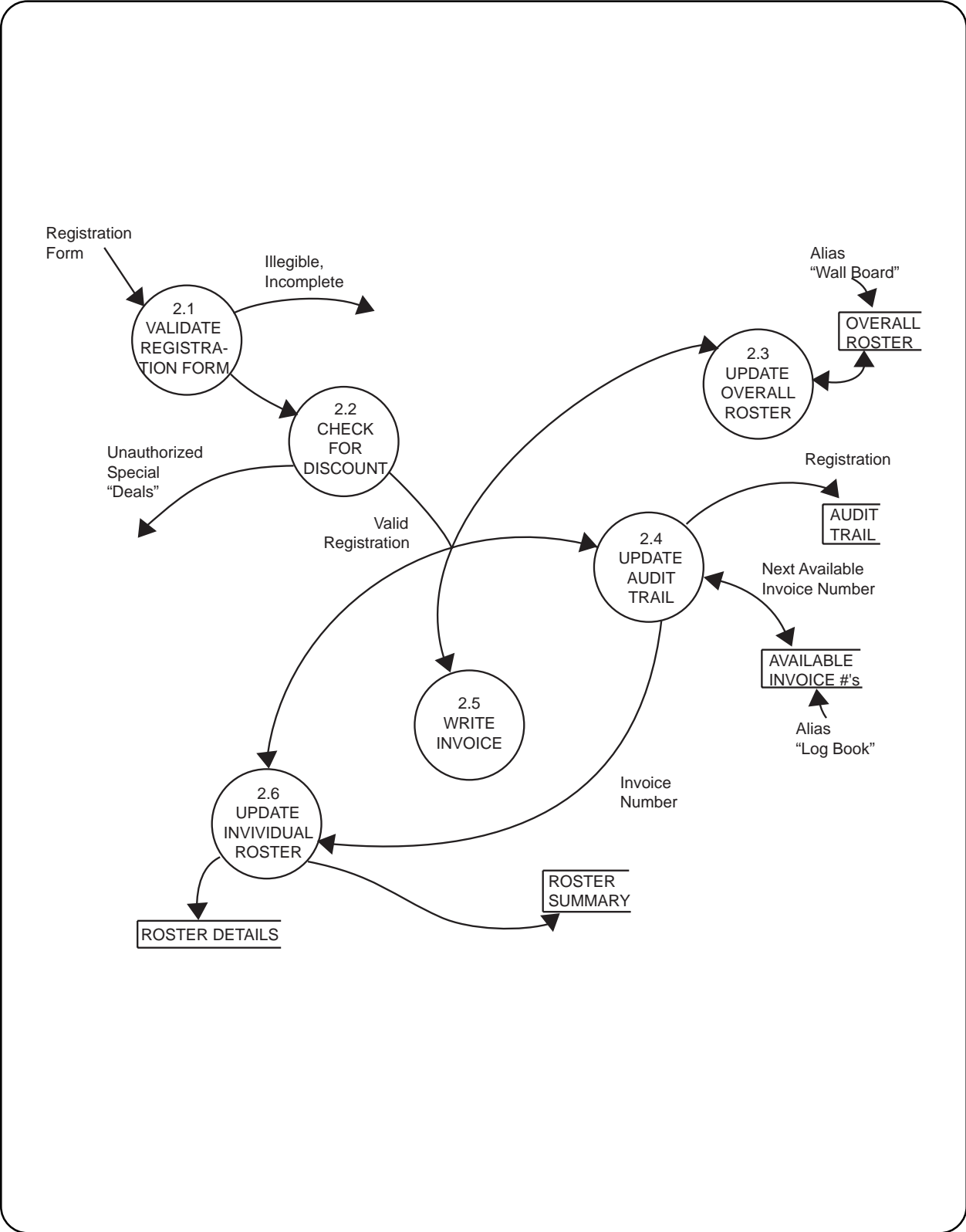


Figure 2.1.5.3.g. A semi-logical data flow diagram. (adapted from Yourdon, p. 76)



**Figure 2.1.5.3.h.** A logical data flow diagram. (adapted from Yourdon, p. 77)

### 2.1.5.4. EXERCISE ON ANALYZING INFORMATION FLOWS.

---

#### **Explanation**

In Module 1.1.18.10., you did a context diagram for a domain of responsibility. The context diagram is a zeroth-level DFD. You can partition the domain of responsibility to produce more detailed DFD's. Your objective is to continue to partition domains and subdomains until you find every information flow in the organization. When you know each information flow, you can figure out the data carried along with each piece of information. The name of the game is to find the data requirements so you can find out how to acquire, store, retrieve, and manipulate only the needed data to later make information through comparing indicator data to reference data.

You want to use management system analysis. Start by surveying the work and knowing and delimiting the domain of responsibility. Then you can find the decisions made to manage the work flow. With the information flows, you can get to the data you need so you can figure out what measurements to make of the work flow to ultimately run through management tools to support decision making.

#### **Situation Description**

Sally and Bob graduated from Virginia Tech together five years ago. Sally, an engineering graduate, has been successful in technical sales for a major chemical company. Bob, a business graduate, has been an administrative officer for a small company.

Based on their success in working for others, they both wanted to go into business for themselves. They bought a small shoe store in Blacksburg, Virginia, close to their alma mater.

Bob and Sally agreed that Bob would invest 10% more than Sally and thus be the controlling partner in the business.

Sally does the inventory and customer end of the business and Bob does the purchasing and financial end of the business. Sally hired John to carry much of the day-in-day-out customer service. John has a flair for decorating and advertising.

Sally and Bob want to get their management started right. You've been hired as a management consultant to advise them.

#### **Exercise**

Partition the domain of responsibility of the shoe store. Start with the context diagram from Module 1.1.18.10. Retain all external agencies as you move to higher-level, more-detailed data flow diagrams. Draw a level-1 DFD for the shoe store. Then draw a level-2 DFD for any one of the conversion processes in the level-1 DFD. Have you yet reached the level of detail you need for that conversion process so you could get your hands on the data involved?

