Chapter 9: Program Design

# Teaching Tips and Strategies *(from Barbara Wixom)*

The main point of this chapter is to understand the importance of using good program design (and what good design means). I spend a lot of up-front time driving home the importance of creating a maintainable design… and how tough this can be when teams are under the gun to get the system designed and coded ASAP.

I spend two class sessions on this chapter. During the first session, I introduce the structure chart. You can use the applying concepts at CD Selections section to step through one as a class, after first presenting the components in a short lecture. Then, I recommend using Minicase 1 to have the students get hands-on experience. I usually combine this session with teaching students how to build a structure chart using the CASE tool.

I let students get some hands-on experience with design guidelines before I present it in the classroom next session. I have them do some end of chapter questions on cohesion, coupling, fan-in, and fan-out before the next class.

During the second class session, I start off by talking about good design. We go over student answers to the design guideline questions that were assigned.

Next, I present the program specification, and we talk about how the structure chart integrates with program specifications (e.g., one spec for each module, couples matching to program inputs and outputs, etc.). I also try to communicate how good design in the structure chart has to also exist in the program specification. I use the peanut butter and jelly example from below to communicate the importance of good program specifications. It works every time! I explain how large projects could have people rolling on and off the project (especially if staffed with consultants), so good, detailed specifications are vital.

I also make sure students understand that even if they never create a structure chart or write a program specification, they need to understand the underlying design strategies that these techniques uphold.

# War Stories *(from Barbara Wixom)*

## Peanut Butter and Jelly

Professor Mark Huber uses a fantastic exercise in his SAD class to demonstrate the importance of writing clear program specifications…

Bring to class the makings of a peanut butter and jelly sandwich: loaf of bread in package, peanut butter, jelly, knife, and napkin. Then, ask the students at the beginning of class to write out explicit instructions telling someone how to make a peanut butter and jelly sandwich. During a break, go through the instructions and pick out several directions that are easy to misinterpret. Then, when the students are back in the room, tell the students that you have never made PB&J before and will do so using the directions that they created.

Perform the directions literally. For instance, if they say “put peanut butter on bread”, then you can take the entire unopened jar of peanut butter and put it on top the wrapped loaf of bread, etc. It becomes very funny – and a visual example of how easy it is to misinterpret someone else’s directions.

Morale: developing good programming specifications is tough. Many times programmers and designers never meet – and it is up to the person writing the specification to make sure that the programmers know what needs to be done. Often people are moved onto new projects, so a programmer may not have the luxury of being able to clear up ambiguity easily.

# Answer to Your Turn 9-1: Structure Chart

Student answers will vary. The additions to the structure chart include: 2.1 Notify Pilots of New Flight Request and its submodules, and 2.2 Pilot Submits Bid and its submodules. Most of the modules that students include will be transform structures.

# Answer to Your Turn 9-2: Program Specification

Student answers will vary. One solution:

##### Program Specification

Module 2.3.3

Name: Assess Pilot Drone

Purpose: Verify that drone has the capabilities to provide all requirements of the flight

Programmer: Joe College

Date Due: 2/1/19

C PowerScript HTML/PHP VISUAL BASIC

Events

|  |
| --- |
| Bid window has closed for flight and flight is being assigned to specific pilot |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Input Name: | Type: | Used by: | Notes: |
| Flight Request | Record | Program 2.3 |  |
| Flight Bid | Record | Program 2.3 |  |
| Drone | Record | Program 2.3 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Output Name: | Type: | Used by: | Notes: |
| Drone\_OK\_Flag | Boolean | Program 2.3 |  |

Pseudocode

|  |
| --- |
| (Assess\_pilot\_drone module) |
| Drone\_OK\_flag = true (drone is acceptable) |
| Retrieve drone details for drone listed in Flight Bid record |
| For each flight requirement in Flight Request |
| Verify drone has feature needed for specific flight requirement |
| IF drone feature not sufficient for flight requirement, THEN Drone\_OK\_Flag = false |
| Next Flight Requirement |
| Return Drone\_OK\_Flag |

# Answer to Concepts in Action 9-A: Winning by Design

The main reason that E&Y won is that they spent the time to plan and design the system before any programming was done. Typically, more time spent on planning and design results in a detailed understanding of what needs to be developed. Therefore the finished product closely matches the specifications.

Less time spent in the planning and design typically results in greater time spent in programming. Without a detailed specification, errors and/or deficiencies tend to occur that need to be dealt with, thus more programming time is needed.

# Solutions to End of Chapter Questions

1. *What is the purpose of creating a logical process model and then a physical process model?*

During the analysis phase, the logical process models are used to depict the processes and data flows that are needed to support the functional requirements of the new system. However, logical process models do not include implementation details, or show how the final system will work. Physical process models include that information, in terms of technology, format of information moving through processes, and the human interaction that is involved.

1. *What information is found on the physical DFD that is not included on the logical DFD?*

The physical DFD includes all elements on the logical DFD plus: implementation references for data stores (e.g. type of database), processes (e.g. programs) and data flows (e.g. paper reports, input screens, etc.), human-machine boundaries, any additional system-related data stores, processes, and data flows.

1. *What are some of the system-related data elements and data stores that may be needed on the physical DFD that were not a part of the logical DFD?*

Student answers will vary, however additions are typically related to technical limitations or to the need for audits, controls, or exception handling.

1. *What is a human-machine boundary?*

A human-machine boundary line is drawn in each instance where a human will interact with the system. The boundary can represent a web page, an application. Designing placement of human-machine boundaries includes addressing issues of cost, efficiency, and integrity.

1. *Why is using a top-down modular approach useful in program design?*

Analysts should first take time in the design phase to create a maintainable system. In other words, analysts should create a design that is modular and flexible. To do this, analysts can design programs in a top-down modular approach, using a variety of program design techniques. With the top-down approach, the program design is specified broadly, or at a high-level, and then more details are added that show the components of the program and how they work together. Developing program designs with this approach helps to ensure that efficient programs are written, that the programs work together effectively in the system, and that the system performs as it is supposed to perform.

1. *Describe the primary deliverable produced during program design. What does it include and how is it used?*

At the end of program design, the project team compiles the program design document, including all of the structure charts and program specifications that will be used to implement the system. The program design is used by programmers to write code.

1. *What is the purpose of the structure chart in program design?*

The structure chart shows all the components of code that must be included in a program at a high level, arranged in a hierarchical format that implies *sequence* (in what order components are invoked), *selection* (under what condition a module is invoked), and *iteration* (how often a component is repeated).

1. *Where does the analyst find the information needed to create a structure chart?*

One recommendation for creating a structure chart is to begin with the processes depicted on the logical DFD. Each process on a DFD tends to represent one module on the structure chart, and if leveled DFDs are used, then each DFD level tends to correspond to a different level of the structure chart hierarchy. A structure chart is composed of *modules* (lines of program code that perform a single function) that work together to form a program.

1. *Distinguish between a control module, subordinate module, and library module on a structure chart. Can a particular module be all three? Why or why not?*

A control module contains the logic for performing other modules that are subordinate to it. Subordinate modules are ‘underneath’ a higher-level module in the hierarchy. Library modules are modules that perform tasks in several places in the system; they are reused. It is certainly possible for a module to be both a control module (meaning it controls modules beneath it in the hierarchy) and a subordinate module (meaning it is “underneath” a higher level module). It is also feasible that such a module is also used several places within the overall system and therefore is set up as a library module, although this is rare.

1. *What does a data couple depict on a structure chart? A control couple?*

Data couples represent the movement of data elements or structures between modules. Control couples represent parameters, messages, or status flags that are moved between modules.

1. *It is preferable for a control couple to flow in one particular direction on the structure chart. Which direction is preferred and why?*

It is highly preferable for a control couple to be passed from a subordinate module to a control module. This implies that the subordinate module has found a condition that is passed to the control module to use in determining how the program will operate. If the control module passes a control couple to a subordinate, it implies that the subordinate module has control over the higher-level module.

1. *What is the difference between a transaction structure and a transform structure? Can a module be a part of both types of structures? Why or why not?*

A transaction structure contains a control module that calls subordinate modules, each of which handles a particular transaction. Transaction structures often occur where the actual system contains menus or submenus, and they are usually found higher up in the levels of a structure chart. A transform structure, has a control module that calls several subordinate modules in some sequence, after which something “happens.” These modules are related because together they form a process that transforms some input into an output. Often, each module accepts an input from the module preceding it, works on the input, then passes it to the next module for more processing. In a leveled DFD, the lowest levels usually represent transform structures.

1. *What is meant by the characteristic of module cohesion? What is its role in structure chart quality?*

Module cohesion refers to how well the lines of code within each structure relate to each other. Ideally, a module should perform only one task, making it highly cohesive. Cohesive modules are easy to understand and build because their code performs one function, and they are built to perform that function very efficiently. Typically, you can detect modules that are not cohesive from titles that have an *and* in them, signaling that the module performs multiple tasks.

1. *List the seven types of cohesion. Why do the various types of cohesion range from good to bad? Give an example of good cohesion and one example of bad cohesion.*

*Functional cohesion* occurs when all elements of the module contribute to performing a single task, and this form of cohesion is highly desirable. By contrast, *temporal cohesion* takes place when functions within a module may not have much in common other than being invoked at the same time, and *coincidental cohesion* occurs when there is no apparent relationship among a module’s functions (definitely something to avoid). In *sequential cohesion* output from one task is used by the next. In *communicational cohesion* elements contribute to activities that use the same inputs and outputs. In *procedural cohesion* elements are performed in sequence but do not share data. In *logical cohesion* list of activities, which one to perform is chosen outside of module.

1. *What is meant by the characteristic of module coupling? What is its role in structure chart quality?*

*Coupling* involves how closely modules are interrelated, and the second guideline for good structure chart design states that modules should be loosely coupled. In this way, modules are independent from each other, which keeps code changes in one module from rippling throughout the program. The numbers and kinds of couples on the structure chart reveal the presence of coupling between modules. Basically, the fewer the arrows on the diagrams, the easier it will be to make future alterations to the program.

1. *List the seven types of coupling. Why to the various types of coupling range from good to bad? Give an example of good coupling and an example of bad coupling.*

[Correction: This question should ask for only five types of coupling. There are seven types of cohesion.] There are five types of coupling, each falling on different parts of a good-to-bad continuum. *Data coupling* occurs when modules pass parameters or specific pieces of data to each other, and this is a form of coupling that you want to see on your structure chart. A bad coupling type is *content coupling,* whereby one module actually refers to the inside of another module. *Stamp coupling* occurs when modules pass record structures. *Control coupling* occurs when a module passes a piece of information that intends to control logic. *Common coupling* occurs when a module refers to the inside of another module.

1. *What is meant by the characteristics of fan-in and fan-out? What are their roles in structure chart quality?*

Both fan-in and fan-out refer to the number of subordinate modules a control module communicates with. *Fan-in* describes the number of control modules that communicate with a subordinate; a module with high fan-in has many different control modules that call it. Structures with high fan-in promote the reusability of modules and make it easier for programmers to recode when changes are made or mistakes are uncovered, because a change can be made in one place. In a fan-out structure, one control module communicates with multiple subordinate modules. As the number of subordinate module increases, the level of complexity of communication increases. Ideally, the fan-out structure is more efficient and effective than the fan-out module. When assessing structure chart quality, attention should be paid to a fan-out module to determine whether or not it can be redesigned. The rule of thumb is that one control module should communicate with a maximum of seven subordinate modules.

1. *List and discuss three ways to ensure the overall quality of a structure chart.*

Following structure chart design guidelines produces programs that are modular, reusable, and easy to implement. First, modules should be built with high cohesion. This means that the lines of code within each module relate to each other, and the module performs one and only one task. This makes the modules easy to build, efficient, and easy to understand. Second, modules should be loosely coupled. This means that the modules are independent from each other, so that code changes in one module have minimal impact on other modules. Third, design the structure to create high fan-in and avoid high fan-out. High fan-in implies that a module is called from several places within the structure, meaning that it is reused. Avoiding high fan-out means that we want to minimize the number of subordinate modules associated with a control module. Generally, a control module should have no more than seven subordinate modules.

1. *Describe the purpose of program specifications.*

Program specifications include explicit instructions on how to program pieces of program code. During the preparation of the program specifications, the analyst may discover design problems in the structure chart, or may find better ways to arrange the modules.

1. *What is the difference between structured programming and event-driven programming?*

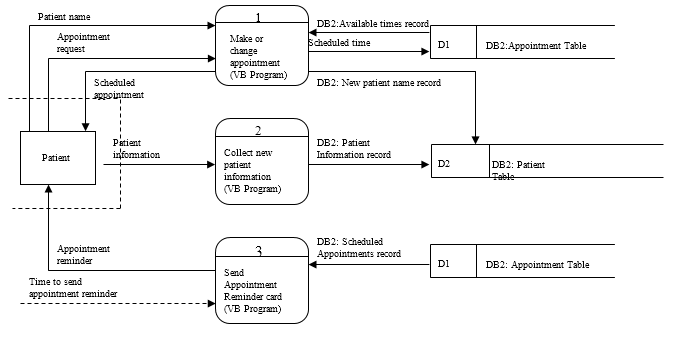
Structured programming involves writing programs and procedures that are executed in a strict order by the computer system, and users have no ability to deviate from that order. Event-driven programs include procedures that are executed in response to an event initiated by the user, system, or program code. After initialization, the program waits for some kind of event to happen, and when it does, the program carries out the appropriate task, then waits once again. .

1. *Is program design more or less important when using event-driven languages such as Visual Basic?*

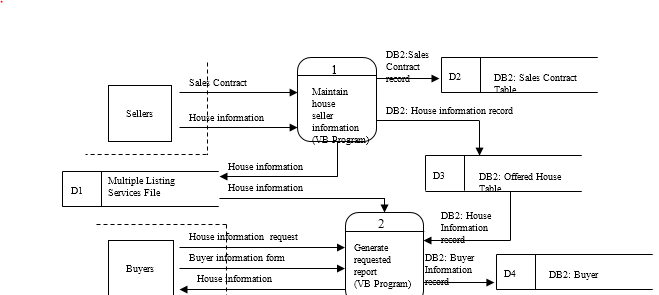
Program specifications may be used when programming in event-driven languages. There are other tools that may be useful in designing these circumstances, such as the state-transition diagram that is associated with UML (Unified Modeling Language) diagrams.

# Solutions to End of Chapter Exercises

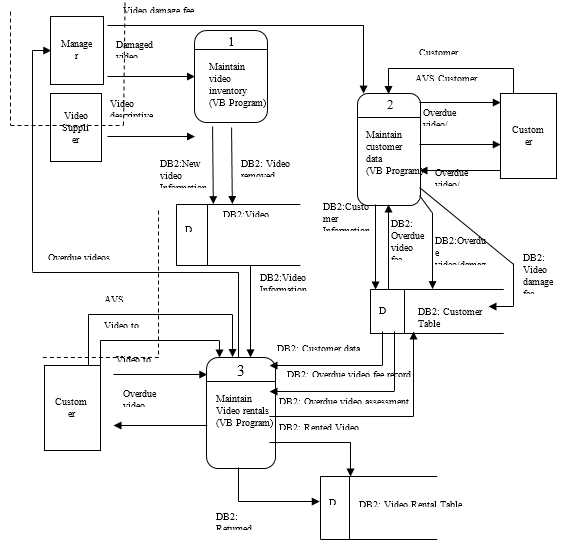
1. *Draw a physical level 0 data flow diagram for the following dentist office system…*



1. *Create a physical level 0 data flow diagram for the following AREI system…*



1. *Draw a physical level 0 data flow diagram for the following AVS system…*

**

1. *What symbols would you use to depict the following situations on a structure chart?*
   * *A function occurs multiple times before the next module is invoked.*

A curved arrow around the module branch

* + *A function continued on the bottom of the page of the structure chart.*

An on-page connector (a circle)

* + *A customer record is passed from one part of the program to another*

A data couple is used (arrow with an open-ended circle at the end)

* + *The program will print a record either on screen or on a printer, depending on user’s preference.*

The branches to the two modules starts from a diamond symbol

* + *A customer’s ID is passed from one part of the program to another*

A data couple is used (arrow with an open-ended circle at the end)

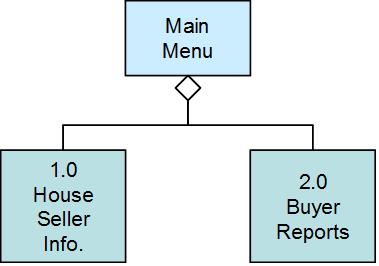
* + *A function cannot fit on a current page of the structure chart*

Use an off-page connector to link to the function on a separate page.

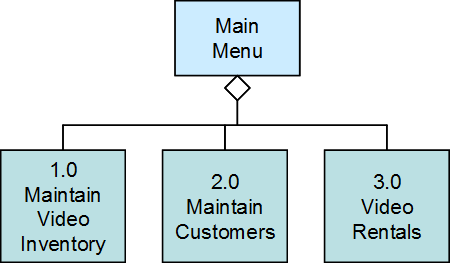
1. *Describe the differences in the meanings between the following two structure charts below. How have the symbols changed the meanings?*

Structure chart (a) indicates that first a book is found by its author, then a report is printed (header, detail, footer). Structure chart (b) shows that the module to print the report header may be called, then a series of repetitive calls will be made to find a book by author and print the report detail, until an out of stock condition is reached. The module to print the report footer may then be called. The arc in chart (b) indicates a repetitive process, and the diamond symbol indicates that one or the other module will be called.

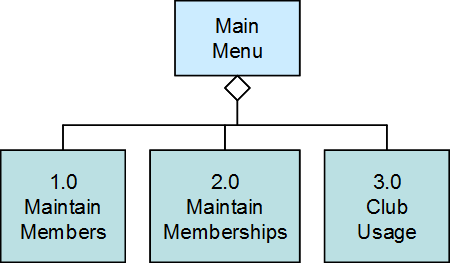
1. *Create a structure chart based on the DFDs that you created for the following exercises in Chapter 4:*
   * *Question D*



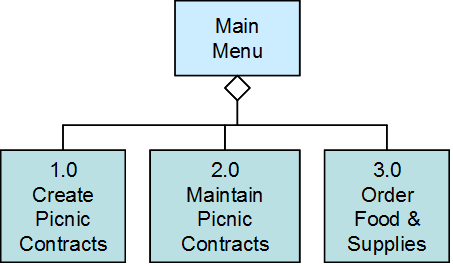
* + *Question E*



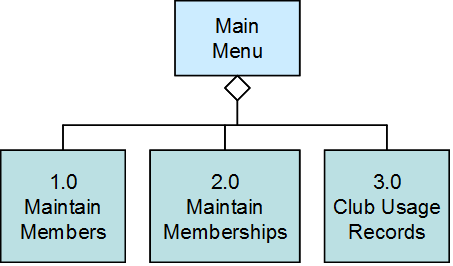
* + *Question F*



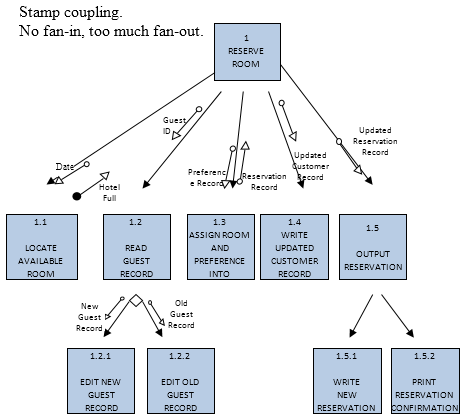
* + *Question G*



* + *Question H*



1. *Critique the structure chart shown, which depicts a guest making a hotel reservation. Describe the chart in terms of fan-in, fan-out, coupling, and cohesion. Redraw the chart to improve the design.*



1. *Identify the kinds of coupling that are represented in the following situations:*

(a) stamp coupling (modules pass record structures)

(b) stamp coupling (modules pass record structures)

(c) stamp coupling (modules pass record structures)

(d) data coupling (modules pass fields of data or messages)

1. *Identify the kinds of cohesion that are represented in the following situations:*

Communicational; procedural; coincidental

1. *Identify whether the following structures are transaction or transform and explain the reasoning behind your answers.*

(a) transaction – the control module calls one of the subordinate modules, each of which handles a particular transaction.

(b) transform – each subordinate module is called in sequence by the controlling module to complete the transformation of some input into an output.

(c) upper half of diagram is transaction; the controlling module calls one of the subordinates to perform a particular transaction. The lower half is transform; the modules are called in sequence to complete a transaction.

1. *Create a program specification for module 1.1.3.1 on the structure chart in Figure 9-12.*



1. *Create a program specification for module 1.2.3.4 on the structure chart in Figure 9.12.*



1. *Create pseudocode for the program specification that you wrote in Exercise K.*

See Pseudocode section of Exercise K above

1. *Create pseudocode for the program specification that you wrote in Exercise L.*

See Pseudocode section of Exercise L above

1. *Create pseudocode that explains how to start the computer at your computer lab and open a file in the word processor. Exchange your pseudocode with a classmate and follow the instructions exactly as they appear. Discuss the results with your classmate. At what points were each set of instructions vague or unclear? How would you improve the pseudocode that you created originally?*

Student responses will vary depending upon the login procedures at their local lab and the word processing software available.

# Answers to Textbook Minicases

1.a. Develop a structure chart for this segment of the Holiday Travel Vehicles system.



1.b. *This is a transform structure chart. All of the modules shown are called in a sequence in order to produce the desired output. This is at a very low level in the system hierarchy, which is typical of transform structures.*

2. Develop a program specification for Module 4.2.5 (Calculate Dealer Cost) in Minicase 1.



# Supplemental Minicases

1. Assume you are the project manager on a large systems development project team. Your team has been working on a major new system for your company. The project has progressed very well, and your team is about to begin program design activities. Because of the project scale, you are managing a large team, and you have organized the staff into four programming teams with four team leaders. Your team leaders are experienced analysts, and you will be relying upon them to prepare the program designs for the parts of the system that you assign them.

One of your concerns is that despite their experience, your programming team leaders have not worked together before, nor have any of them worked for you on any previous projects. To prepare for the program design work, you have decided that it will be important that you establish some guidelines for the development of the program structure charts. Develop a set of guidelines that will be helpful to the team in assuring consistency in their structure charts.

*Answer:*

1. *Design the modules so that each module performs one and only one function or task.*
2. *Modules will be passed only the information that is actually needed in the module.*
3. *Limit the amount of information shared by modules; make them as independent as possible.*
4. *Develop and use library modules whenever feasible to maximize reusability. Design the structure charts with high fan-in.*
5. *Limit the complexity of control modules by having them control no more than seven subordinates.*
6. *Always pass control couples from low to high; avoid passing control couples from high to low.*
7. *Strive for reasonable size in the modules, neither too large nor too small.*
8. As lead analyst on a new system development project, you have been hard at work developing the program designs for the new system. Much of your time over the last three weeks has been devoted to this task, and you are nearing your target deadline. You have been so immersed in this work that you have decided to ask a team member to spend some time reviewing the structure charts that you have prepared. Although you have tried to be very careful in your work, you know that it is easy to overlook some design problems, and you want to avoid as many nasty surprises later as you can. The team member you want to help you has had quite a bit of experience reading and implementing systems from structure charts, but she has not been responsible for their development yet in her career.
9. Describe what your team member should look for with regard to cohesion. What will signify good cohesion in the design? What should she look for that might signify poor cohesion in the design?

*Answer: The key to evaluating the structure charts regarding cohesion is to look at the work that is performed in each module. Ideally, each module should perform one and only one problem-related task. As you check the modules in the structure chart, look for situations where more than one thing is done in the module. Pay particular attention to modules that might do several things that are unrelated. That is the worst type of problem, and should be corrected right away.*

1. Describe what your team member should look for with regard to coupling. What will signify good coupling in the design? What should she look for that might signify poor coupling in the design?

*Answer: The key to evaluating structure charts with regard to coupling is to study the data and control couples that have been specified on the chart. The best design will pass only the specific data or control messages that are needed by the receiving module. If more information has been passed than the module needs, we should correct that. Also, make sure that control information primarily goes from ‘low to high’ in the charts. If control couples are passed from high to low, there should be some modifications made. Finally, try to make sure that the module have a minimal amount of interaction with each other.*

1. Describe what your team member should look for with regard to fan-in and fan-out. What will signify high fan-in in the design? What should she look for that might signify low fan-in in the design? What will signify low fan-out in the design? What should she look for that might signify high fan-out in the design?

*Answer: High fan-in is good, and implies that modules that do the same thing have been identified and set up as library modules. If you see any places where the same function is performed by several different modules, we may be able to set that up as a library module and achieve higher fan-in. Fan-out refers to the number of subordinates that have been set up under a control module. If you find a situation where there are more than seven subordinates under a control module, then that should be reorganized to reduce the amount of fan-out.*

# Experiential Exercises

1. Purpose: to increase understanding of structure charts.

Have students develop an outline of a transaction system such as their favorite ATM system. Based on their notes, have them ‘work backwards’ from the menus of the ATM and develop a structure chart for the upper levels of the system. Point out the transaction structure at the top levels of the system. If the students look at different ATM terminals, have them compare the different structures that have been implemented in each. Discussion: Can the students develop ideas as to why the different structures were chosen?

1. Purpose: to increase understanding of structure charts.

Divide the class into small groups. Have students develop an outline of a transaction system such as their favorite ATM system. Based on their notes, have them ‘work backwards’ from the menus of the ATM and develop a structure chart for the lower levels of the system. Have each group develop a structure chart for one of the transactions. Point out that these are (probably) transform structures.

1. Purpose: to become more familiar with structure charts.

Bring an example of a structure chart from an actual system. It may be possible that the IS staff on campus will be willing to share. Distribute the structure charts to the class, and discuss the structure used in the system. Although it may be difficult to tell, how well does the example conform to the ideal of high cohesion, loose coupling, and high fan-in?

1. Purpose: to become more familiar with program specifications.

Bring an example of program specifications from an actual system. It may be possible that the IS staff on campus will be willing to share. Distribute the example to the class. Discuss the components included in the program specs. How close does this example come to the example in the book? Would the students be able to develop a program from the example? What else (if anything) would they like included?