Complete Systems Analysis

James & Suzanne Robertson

foreword by Tom DeMarco

The Workbook, The Textbook, The Answers

Complete Systems Analysis

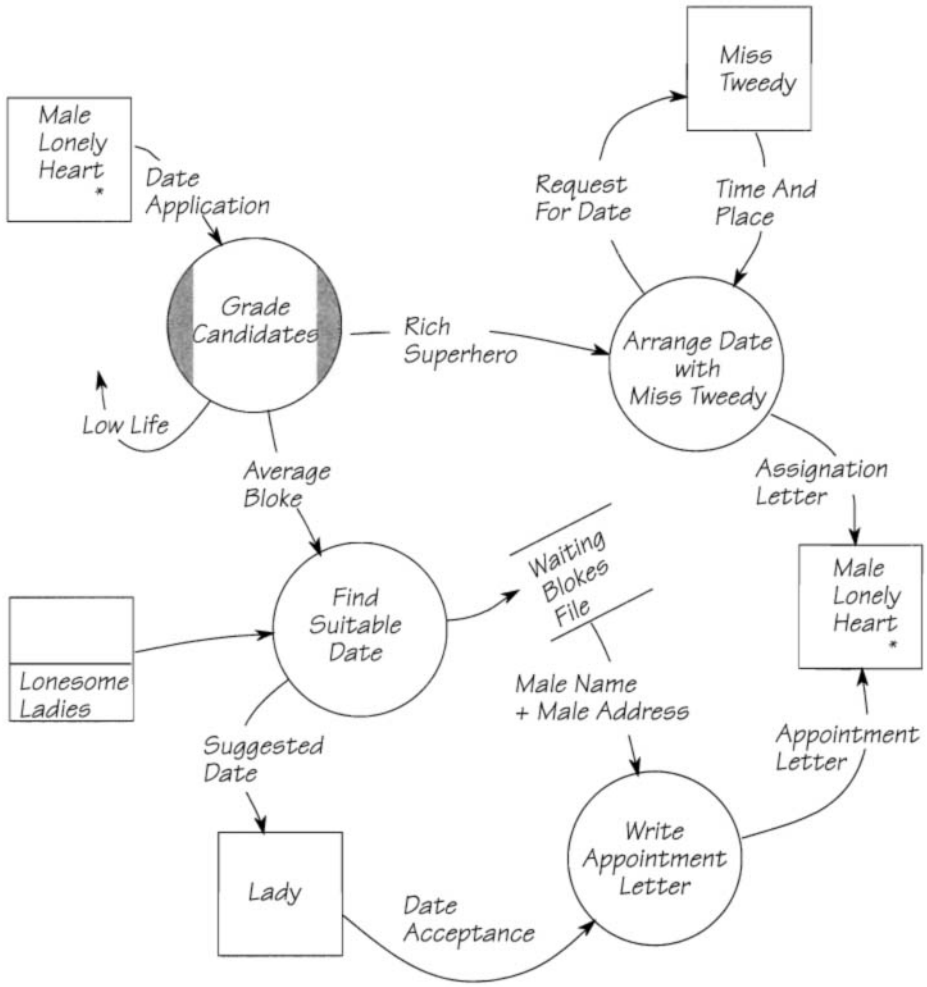

Figure 2.6.1: A data flow diagram of Miss Tweedy's Dating Service. Miss Tweedy runs a dating service, but she'd rather marry a rich man and retire. She has left instructions that if any eligible men apply, they are to be referred directly to her.

This system exists to provide data to the outside world. The lonely ladies expect that the system will provide them with information concerning suitable men who wish to meet them. We can see this expectation represented by the data flow SUGGESTED DATE and the terminator LADY. Miss Tweedy also has expectations that the system will supply her with a husband.

You can say this is a working model of the system provided you can demonstrate that the model produces the correct output when it is given the expected input. This correctness convention is called the *Rule of Data Conservation.*

The Rule of Data Conservation

In Figure 2.6.2, for example, the process FIND SUITABLE DATE needs the data flow AVERAGE BLOKE to contain enough data to find a suitable pairing in the LONESOME LADIES data store. The combination of data from the flow and the store is used to generate SUGGESTED DATE and an entry in the WAITING BLOKES FILE.

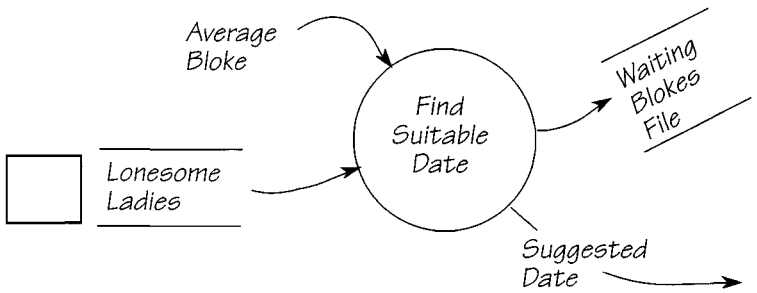

Figure 2.6.2: *The Rule of Data Conservation states that each process in the data flow diagram must be able to produce the output data flows from its input.*

Naturally, you would find the rule easier to demonstrate if you had first defined the data content of the flows and stores. Chapter 2.9 *Data Dictionary* discusses how to define the data, but for the moment, just look at the outputs and think about what data is needed in the incoming flows to make this process work. Along with the incoming data, a process may also use internal calculations and any constant information contained within the process to construct its outputs. Additionally, a process often needs to have access to the current time, day, and date. Rather than repeating clocks and calendars all over the model, you may assume that each process knows the current time, day, and date.

Each process must receive data that are both sufficient and necessary to construct its output.

In Figure 2.6.2, the output flow to the WAITING BLOKES FILE would contain almost identical information to the AVERAGE BLOKE flow. The SUGGESTED DATE flow would contain the lady's name and address (from the LONESOME LADIES data store), and the name of the man (from AVERAGE BLOKE). There is nothing to suggest that this process breaks the Rule of Data Conservation.

When you can prove that all the processes in the model can construct their output from their input, you can say your model is a working model.

Triggering Processes

A process can be active only when it has data to work with, so it must wait for the data flow to deliver the necessary data. So we say the data flow triggers the process. In Figure 2.6.3, DATE ACCEPTANCE triggers the process WRITE APPOINTMENT LETTER. At some stage, the process reads the data store WAITING BLOKES FILE for additional information to construct the outgoing flow.

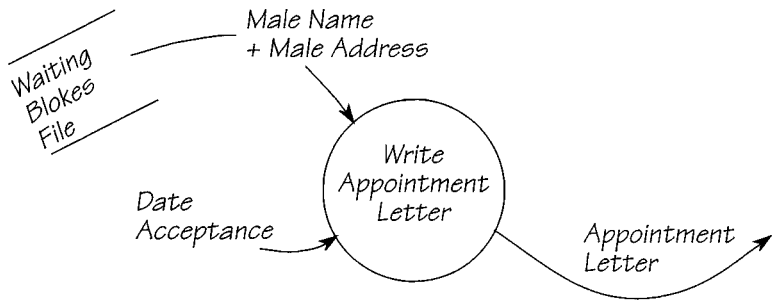

Figure 2.6.3: A process becomes active when the incoming data flow arrives.

Since a data flow makes a process active, it follows that the production of data controls the system's activity. That is, the system's operation is regulated by the processes themselves as they manufacture data for each other. There is no external entity directing which processes are to be active and which should remain inactive.

This makes the data flow diagram a natural way of representing the system. After all, in an office, you don't see the office staffers standing around waiting their turn to become active. They start work as soon as they have data to process. (At least, this is how the office staff should behave.)

Also, consider what you are trying to do. Your task is to define the requirements of the system. These requirements must not contain elements that are part of the current control or management mechanism. You do not know during the analysis stage if these mechanisms are appropriate for any future implementation of the system. If you don't yet know *what* the system has to do, you have no way of knowing *how* it is going to do it.

The data flow diagram is a model of the system that contains only the data and functional requirements. It excludes all forms of control mechanisms.

Naming Data Flows

Every data flow must have a name. Choosing a name is important, and the name should be self-explanatory so that it contributes to the reader's understanding of the system. Because the data flow diagram is a model of some reality, the data flows exist in the real world. For this reason, data flows are usually easy to name—you just use their real-world name. You want names that communicate the meaning of the data. Test your data flow names by asking the question, "Does this name relate to the subject matter of the system, or is it concerned with how the data flow is implemented?" Sometimes, depending on the answer, you will need to rename the data flow.

For example, we worked on a system in which the users referred to a data flow called GREEN REPORT. When we asked what about the report made it green, we found out that the report was printed on green paper, but it actually contained information about debtors. Since the name did not communicate the subject matter of the data flow, we changed the name, with the users' agreement, to DEBTORS REPORT.

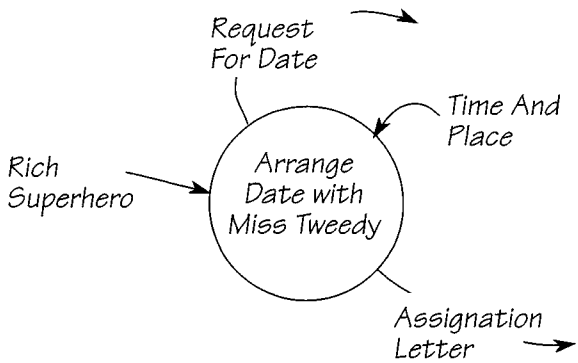

Figure 2.6.4: *Each data flow has a name that lets the reader easily comprehend that part of the system. The name should be one that is well-established in the users' business or, failing that, is a suitable invention by the analyst.*

Always name the flows in your models *before* naming the processes. Try this as a test: Take any data flow diagram and put coins or something over the processes to hide their names. Does the model still make sense? It should, provided the data flows have been well-named.

The processes are there to transform the data flows. If all you can see of your system model are the data flows, the processes should be fairly self-evident. Now uncover the processes and this time conceal the

Give all data flows a name that is both self-explanatory and recognizable.

names of the data flows and stores. Does the model still make sense? Not as much. The reason is that the processes are too general without the data flows to give them context. The point of this exercise is that if you do not understand the data, the processes make little sense.

Composite Data Flows

Use composite data flows when you want to have your cake and eat it, too. Let us explain. Suppose you have a data flow containing several data elements, or groups of elements. The most common way of dealing with this situation is to name the data flow and then define its composition in the data dictionary. This is a perfectly reasonable way to control complexity, especially in dealing with complex flows.

Another solution, which avoids the entry in the data dictionary, puts the names of the data elements directly onto a single, composite data flow.

Figure 2.6.5 demonstrates a composite data flow, consisting of two data elements. Each of the names attached to the flow is a recognizable element in its own right, and has its own definition in the data dictionary. By using this idea, you reduce the number of entries in the data dictionary.

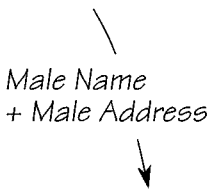

Figure 2.6.5: A composite data flow showing its components.

Data Flows Are Always Named, Except ...

A data flow without a name is like a millionaire without money. However, there is an exception. You may omit the name when the flow is to or from a data store and the data flow has exactly the same content as the store. In other words, the flow carries a hundred percent of the store's data elements. Since the store must be defined in the dictionary, it would be redundant to name and define the flow. Also, omitting the name makes the model more readable.

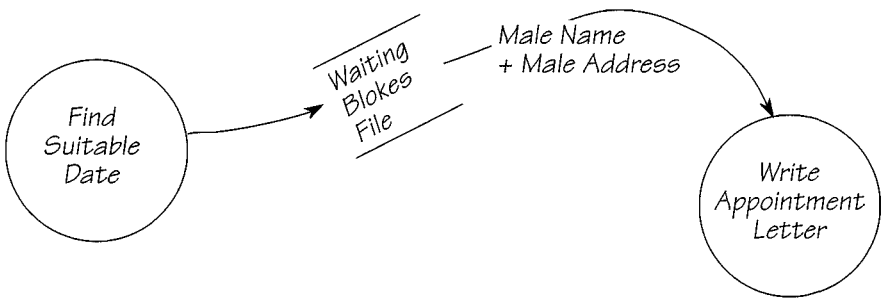

Figure 2.6.6: The process FIND SUITABLE DATE writes all the data in the store WAITING BLOKES FILE. In this case, there is no need to label the data flow.

However, sometimes the flow and the store are not the same. For example, in Figure 2.6.6, the flow coming from the store carries a name because it is a unique collection of data elements from the store. Later on, when you build essential requirements models, you will see examples of an exception to the rule for naming data flows.

Data Flows and Data Stores

A data flow represents data moving from one place to another. A data store is a static container of data. Data flows are deposited in the store for later retrieval.

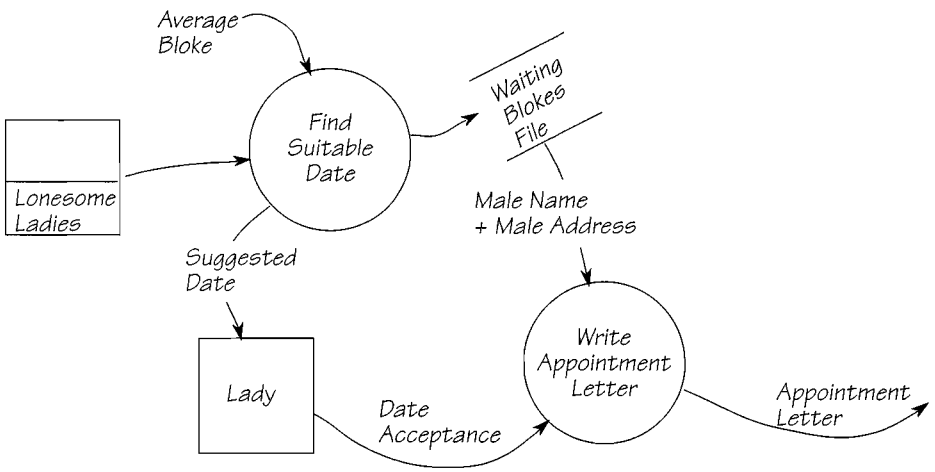

Figure 2.6.7: The system remembers the data in the store WAITING BLOKES FILE *and the system uses the data when it is time to produce an* APPOINTMENT LETTER.

The data store indicates a time delay between the storage and the retrieval of the data. Before showing a data store on your model, be certain that the time delay is necessary for the system to carry out the business policy. You'll often come across data stores that have no policy-related reason to exist. An example of this is shown in Figure 2.6.8.

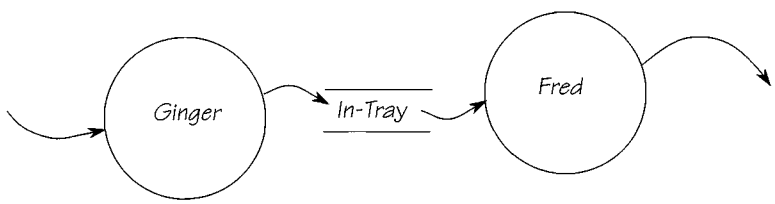

Figure 2.6.8: The IN-TRAY *is not a real data store. It exists because one processor (*FRED*) is not fast enough to keep up with the incoming data from* GINGER*. There is no business policy reason for this data store.*