

## FEATURE ARTICLE

George Martin

# Structured Design

## Part 1: An Introduction to Structured Techniques

Blurb ????



Sometimes I start to write an article, and I find myself at a loss for words. Well, I had no such problems this month. I'd like to present a concept called structured design (or structured programming) and, in particular, how it pertains to flowcharts. Perhaps you've heard of structured design or have taken a course or two about it. My career was timed just right so that I ran into complicated software projects at the same time a lot of work was developed with structured techniques. It was helpful and stuck with me. In case you've never come across this or have forgotten, let me refresh your memory.

In the late '60s and early '70s, programs started becoming so complex that it was increasingly difficult to understand them. Where we once had paper, tape, and card decks, there was now floppy disk program storage. Writing and editing large programs

was made easier, and the number of lines of code a programmer could control soared. However, these larger programs were ill conceived and poorly implemented. Jumps within the program were so frequent that logical program flow was not obvious and debugging was nearly impossible, often referred to as spaghetti code. To overcome this situation, software engineering was developed and structured programming was born.

One key principal of structured programming in a structured design is that every program can be created with a limited number of structural elements. Some experts say that all you need is sequence, selection, and iteration, and I agree. Let's look at these elements.

### SEQUENCE

First, let's take a look at the sequence structure. The sequence of operations is A, B, and C (see Figure 1). This sequence has one entry point and one exit, a key premise of structured design. The operations A, B, and C may be complicated routines, but testing is straightforward and can be completed with a certainty that you've done it correctly.

Many times A, B, and C are referred to as processes. These processes can be as simple as performing an arith-

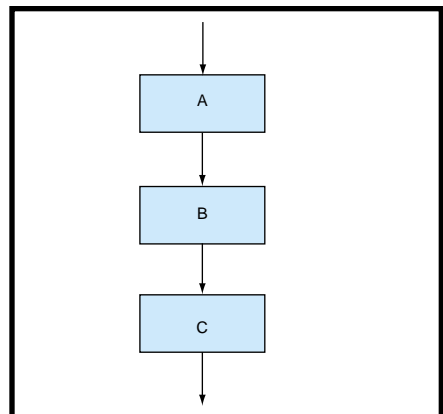


Figure 1—Here you can see the sequence of operations A, B, and C.

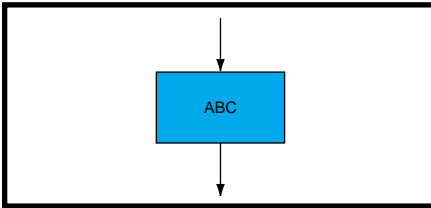


Figure 2—The sequence of operations A, B, and C can be referred to as process ABC.

metic operation ( $a = b + c$ ) or as complicated as `RepaintTheUsersScreen()`. Also, they can be referred to as a single process. Let's call it process ABC (see Figure 2).

You can divide and conquer, break it down into modules, or structure that design. You have a building block that can start small, be tested, and grow to unlimited size and complexity. As I describe different types of structures, notice that they all have one entry and one exit. All the structures can be thought of as or reduced to a simple sequence structure, which is completely understandable and testable.

## SELECTION

In order to understand the selection structure (see Figure 3), you need to know what a condition is. Think of it as a test. For example, is A less than 10? Or, is the A/D converter complete? Those are conditions.

If the condition is true, you execute process E. If the condition is false, you execute D. Note that both D and E do not need to be present, either can be missing. If both are missing, then the test is not necessary.

Again, this structure has only one entry and exit and is still completely testable.

## ITERATION/LOOPING

The third structural element is iteration, or looping. In this situation, the condition is tested, and if it is true,

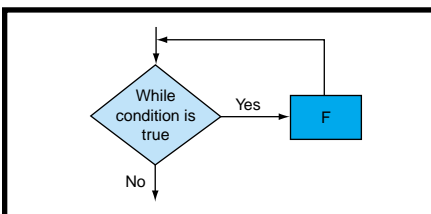


Figure 4—When the condition is false, the program continues. This is called a while loop.

then process F is executed and the test is repeated. If the condition is not true, then the program continues. This is commonly called the while loop (see Figure 4).

A slight variation in the construction of this structure is called the until loop, which can be seen in Figure 5.

Now, let's look at what we have. I contend that all of your code can be written using the three structures I just defined. You can define more structures, such as a case or switch structure, which may make your design more compact or understandable. There are provisions in most languages for this. BASIC has its case statement, and C has its switch statement. But, you can construct either of those out of the three structures shown in this article.

## TESTING

Let me take a break here to talk about testing and testability. I stated that a procedure with a single entry and exit is testable. I hope you find that obvious. You can insert a breakpoint at the start of the procedure, set up all the variables used in the procedure, and then run the procedure until the exit is encountered. This can be repeated until all sets of variables are explored. Because nothing but the variables affects the procedure, you will have completely tested it.

If you have a large complex routine, break that into several small blocks and test each block. Using only these three structures, all the blocks are testable. After each block is tested, you can test larger combinations until you've tested the entire complicated procedure. This is called bottom-up testing.

Consider Figure 6. Don't all of your embedded programs look like this?

## NASSI-SCHNEIDERMAN CHARTS

Now you've got all the tools to do structured design. But, I found that by using only this flowchart technique, I didn't produce great structured designs. It was too easy to let the implementation wander.

Then I was introduced to Nassi-

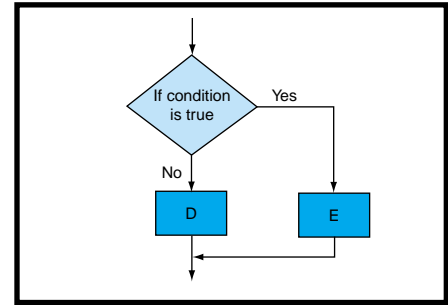


Figure 3—Whether the condition is true or false determines how the selection structure for the process will appear.

Schneiderman (NS) charting. NS charts use the same structured principles I just covered, but they are drawn slightly different. I refused to use these charts in the beginning, feeling that they were too restrictive. But I soon came to realize that this was just the restriction I was looking for. [1-3]

Imagine the entire piece of paper as the flowchart. A vertical line along the left binds the work. Horizontal lines separate sequences or processes.

Does Figure 7 look familiar? These are the three processes charted in Figure 1.

If the condition is true, processes A and B are performed. If the condition is false, then none are performed. Figure 8 shows the selection of Figure 2.

And, the NS charts for the while structure and until structure can be seen in Figures 9 and 10, respectively.

Look over these constructs and check out the web sites listed in the References section. I even included a Java program for producing NS charts on a PC.

Next month, I'll apply these techniques to a flowchart that isn't so structured. It'll be interesting to see

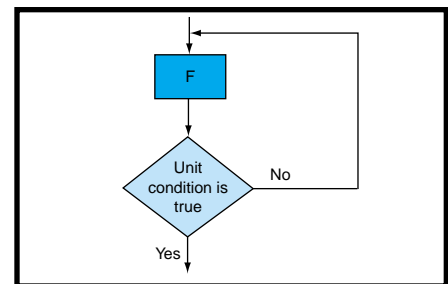


Figure 5—A minor dissimilarity in the construction of Figure 4 is what you see here. This is called an until loop.

what I come up with. 📄

George Martin began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and cofounded a design and manufacturing firm. Typical systems that George designs include servo-motion control, graphical input and output, data acquisition, and remote control. George is a charter member of the Ciarcia Design Works Team and most recently, he's been working on the people-tracking system for Bill Gates' new house. You can reach him at [george.martin@worldnet.att.net](mailto:george.martin@worldnet.att.net).

## REFERENCES

- [1] Nassi-Schneiderman, <http://www.rdrop.com/~cary/html/psd.html>.
- [2] Diagram Generator, <http://www2.informatik.uni-erlangen.de/IMMD-II/Research/Activities/DiaGen/index.html>.
- [3] Structured Programming, <http://www.olympic.ctc.edu/class/akkirkpa/cs165l01.html>, <http://www2.ios.com/~jvn/ch7/ch7.htm>, [http://www.csc.liv.ac.uk/~frans/OldLectures/2CS21\\_Ada/week8/nestedLoops.html](http://www.csc.liv.ac.uk/~frans/OldLectures/2CS21_Ada/week8/nestedLoops.html), <http://www2.ios.com/~jvn/ch2/diagtech.htm>, <http://www2.ios.com/~jvn/ch2/ch2.htm>, <http://www.umsl.edu/~sauter/analysis/dfd/DiagrammingMethods.html>, <http://www.bcvic.net/cs105/unix.html>, [http://www.csc.liv.ac.uk/~frans/OldLectures/2CS21\\_Ada/week6/TopDown.html](http://www.csc.liv.ac.uk/~frans/OldLectures/2CS21_Ada/week6/TopDown.html).

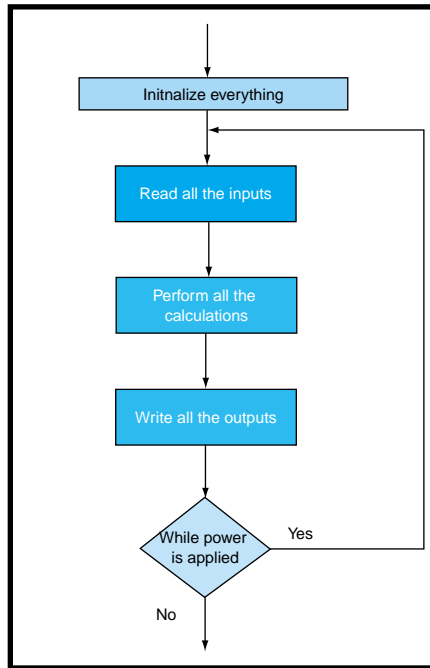


Figure 6—Here is an example of top-down structured designing.

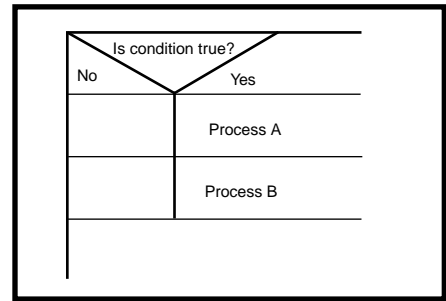


Figure 8—Whether the condition is true or false determines if you go on to the next process.

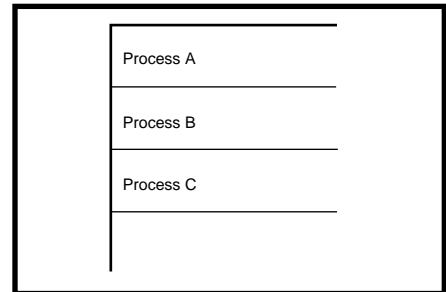


Figure 7—Processes A, B, and C from Figure 1 are seen here in an NS chart format.

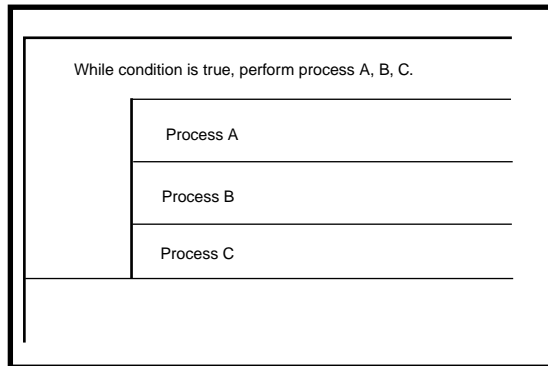


Figure 9—Here you can see the NS chart for the while structure.

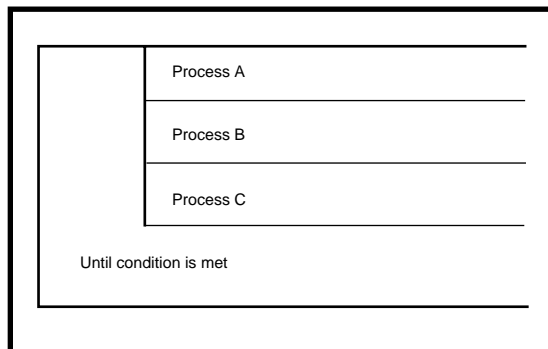


Figure 10—And here, the until structure.

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, [subscribe@circuitcellar.com](mailto:subscribe@circuitcellar.com) or [www.circuitcellar.com/subscribe.htm](http://www.circuitcellar.com/subscribe.htm).