

Knowing the Flow: How Flowcharting Can Help Visualize Software Application Development

Joseph Frantiska, Jr., Ed.D.
Fitchburg State College

Abstract

The complexity of applications that today's computing student needs or desires to create continues to increase. Being able to visualize this complexity can become daunting. Taking a step back in the history of computing allows us to utilize a basic visualization tool that has been around since its early days – flowcharting.

1. Visualization

In this day with tech-savvy students becoming more adept with building increasingly complex applications such as Web sites, multimedia presentations, and animations, students have an increasing need to visualize the design and development of these applications.

Visualization is the act of forming a mental image of something. This can be easy in visualizing a common object such as a ball or book. But how can a student visualize the performance or behavior of a software application prior to its creation?

For the developer, this is critical in being able to develop the application so that it provides the required result. One tried and true visualization method used by systems and software developers for decades is flowcharting.

However, this technique is not just for high-tech types but for anyone, including students, who need to understand the complexity of the application (Web site, program, etc.) they want to construct. It also serves as a good means to explain and document the application's possibly arcane construction and usage to other people in a concise, logical manner. It is the map that describes the route; the book that tells the story.

2. History

The first structured method for documenting process flow, the flow process chart, was introduced by Frank Gilbreth to members of the American Society of Mechanical Engineers

(ASME) in 1921 as the presentation, "Process Charts—First Steps in Finding the One Best Way."

Gilbreth's tools quickly found their way into industrial engineering curricula. In the early 1930s, an industrial engineer, Allan H. Mogensen, began training business people in the use of some of the tools of industrial engineering at his Work Simplification Conferences in Lake Placid, New York.

A 1944 graduate of Mogensen's class, Art Spinanger, took the tools back to Procter and Gamble where he developed their Deliberate Methods Change Program. Another 1944 graduate, Ben S. Graham, Director of Formcraft Engineering at Standard Register Corporation, adapted the flow process chart to information processing with his development of the multi-flow process chart to display multiple documents and their relationships.

In 1947, ASME adopted a symbol set derived from Gilbreth's original work as the ASME Standard for Process Charts. Additionally, Herman Goldstine developed flowcharts with fellow famed mathematician and computing pioneer, John von Neumann, at Princeton University in late 1946 and early 1947 as they developed early electronic computers.

3. Tools

What are the tools that we can use to create a flowchart? A low-tech approach may be to buy a flowcharting template (Figure 1) that can be bought at retailers such as Staples or OfficeMax. There are numerous brands of templates with differing symbols and layouts.

The symbols found on flowcharting templates are designed to provide functionality for a variety of applications (Table 1). Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them.

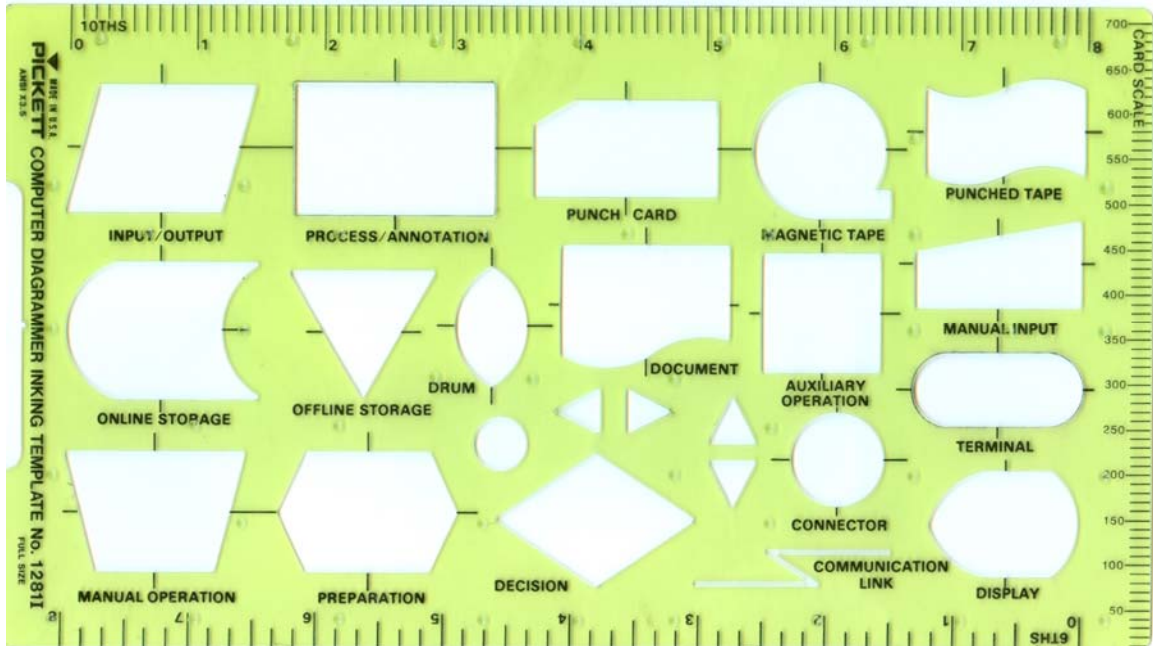


Figure 1. A typical flowcharting template

Table 1
Flowcharting Symbols and Their Meanings

Meaning	Symbol
Start/End The terminator symbol marks the starting or ending point of the system. It usually contains the word "Start" or "End."	
Action or Process A box can represent a single step, or an entire sub-process within a larger process.	
Document A printed document or report.	
Decision A decision or branching point. Lines representing different decisions emerge from different points of the diamond.	
Input/Output Represents material or information entering or leaving the system, such as a student answer (input) or a result (output).	

Connector Indicates that the flow continues where a matching symbol (containing the same letter) has been placed.	
Flow Line Lines indicate the sequence of steps and the direction of flow.	
Data storage Indicates a step where data gets stored.	
Database Indicates a list of information with a standard structure that allows for searching and sorting.	
Display Indicates a step that displays information.	
Off Page Indicates that the process continues off page.	

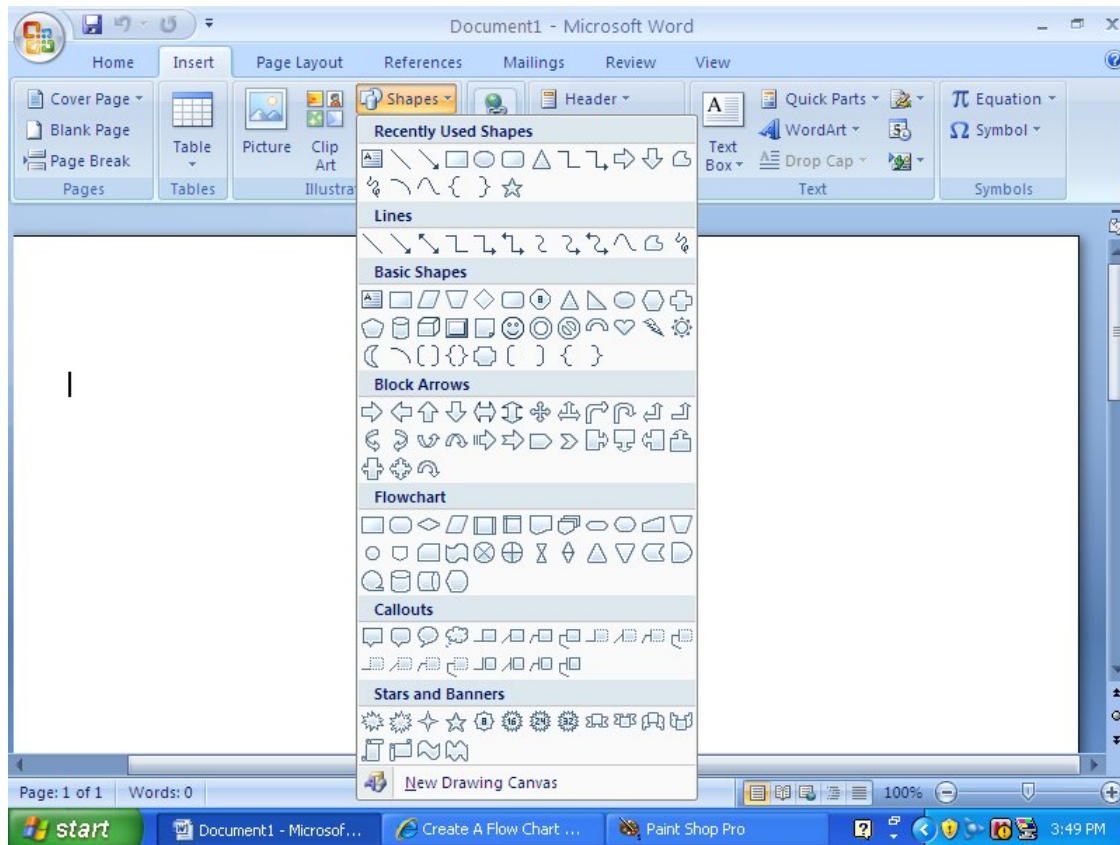


Figure 2. Accessing flowcharting symbols in Microsoft Word

4. Benefits and Implementation

Flowcharts help in the visualization process by providing the following benefits:

1. *They promote process understanding by explaining the steps pictorially.* People may have differing ideas about how a process works. A flowchart can help you gain agreement about the sequence of steps. Flowcharts promote understanding in a way that written procedures cannot do. One good flowchart can replace pages of words.
2. *They provide a tool for training others.* Because of the way they visually lay out the sequence of steps in a process, flowcharts can be very helpful in training other people to use and modify the application according to standardized procedures.
3. *They identify problem areas and opportunities for process improvement.* Once you break down the process steps and diagram them, problem areas become more visible. It is easy to spot opportunities for simplifying and refining your process by analyzing decision points, redundant steps, and other portions.

Implementing flowcharts involves the following steps:

1. *Start with the big picture.* It is best to draw a macro-level flowchart first. After you have depicted this big picture of the process, you can develop other diagrams with increased levels of detail.
2. *Observe the current process.* A good way to start the flowcharting process is to walk through the current process, observing it in actual operation.
3. *Record the process steps you observed.* Record the steps as they actually occur in the process as it is. Write the steps on index card notes. You can use a different color to represent each individual or group involved if that will help you understand and depict the flow more accurately.
4. *Arrange the sequence of steps.* Now arrange the cards or Post-it™ notes exactly as you observed the steps. Using cards lets you rearrange the steps without erasing and redrawing and prevents ideas from being discarded simply because it is too much work to redraw.
5. *Draw the flowchart.* Depict the process exactly as you observed, recorded, and arranged the sequence of steps.

Once you know the proper method of combining the symbols into a flowchart, the process within an application can be communicated via a common language. To understand the basics of flowcharting, examine a very high

level example of what the symbols represent and how they can be combined. In Figure 3 we see a flowchart that delineates a simple process.

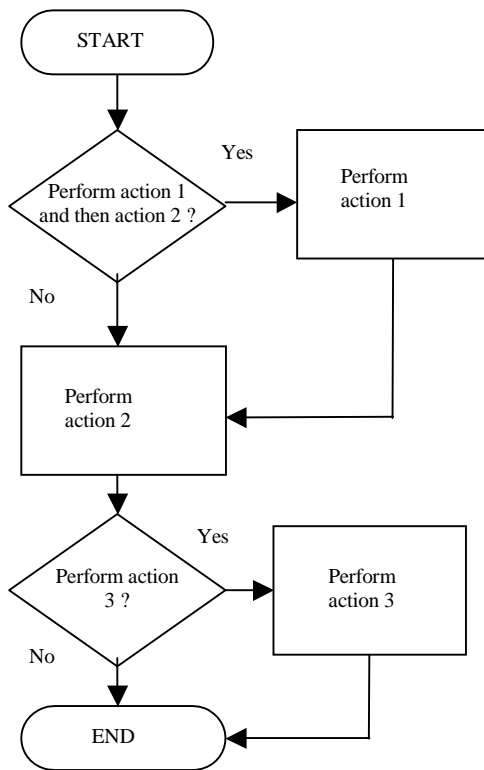


Figure 3. A simple flowchart

First, we draw a start/end symbol and label it “Start.” We then draw a line with an arrowhead showing the direction of process flow. Following that, we begin the actual process by drawing a decision symbol as the destination of the process flow line and labeling it with the question that is to be answered to determine how the processing will flow.

Say that we need to determine if either action 1 will be followed by action 2 or will only action 2 be performed. Branches coming off the decision symbol with an associated answer (yes or no) that will determine if that branch is followed are drawn. That is, if we answered “Yes” to the question, action 1 will be performed and then action 2 will be performed. Otherwise, if we answer “No,” only action 2 will be performed. The next step in the process will be encountered as we are asked if action 3 needs to be performed. If the answer is “Yes,” action 3 will be performed and then the program will end. If the answer is “No,” the program will end.

For a more detailed, real-world example, imagine that you are a K-12 mathematics teacher who wants to use the power of technology to allow your students to develop an application that instructs a person in division. The student

may start by asking, “How can I visualize in a tangible manner, the complexities and interactions of this application?”

You understand that the flow of processing in this application can move along numerous pathways; some can lead to errors and others to a correct solution. The old saying of “measure twice, cut once” certainly applies here. Specifically, try to avoid getting into an error situation instead of working to correct an avoidable situation. In order to do this, we must understand the correct flow of processing along with any possible areas that cause problems and devise methods to avoid these areas.

The portion of the application that is devoted to division has basic rules of processing and error handling. The basic requirements of this application are that the number to be divided is accepted first and it must be an integer. The number that will be the divisor is then accepted and it also must be an integer. Also, it must not be zero.

So what is the basic flow of control in this application? We must begin the application with the ability to accept two numbers and check whether each is an integer. First, the number to be divided (the quotient) is tentatively accepted. Then the number that will divide the quotient (the divisor) is tentatively accepted. As previously mentioned, we must check to make sure that the divisor is not zero since a zero divisor will produce an undefined result. Figure 4 depicts the steps that describe the flow of control of the application also known as an algorithm which are as follows:

1. Accept a candidate quotient value.
2. Check to see if the value is an integer.
3. If not, reject that value, produce an error message, and return to the previous portion of the application to accept another quotient candidate.
4. If the quotient is an integer, accept a divisor value.
5. Check to see if the divisor value is an integer.
6. If it is not an integer, apply the processing of step 3 to the divisor value.
7. Check to see if the divisor value is zero.
8. If it is zero, reject that divisor value, produce an error message and return to the previous portion of the application to accept another quotient candidate.
9. If it is non-zero, perform the operation by dividing the quotient by the divisor.
10. Display the result on the user’s computer screen.
11. Send the result to the printer to produce a hardcopy.
12. Store the result into an existing structure for later examination.

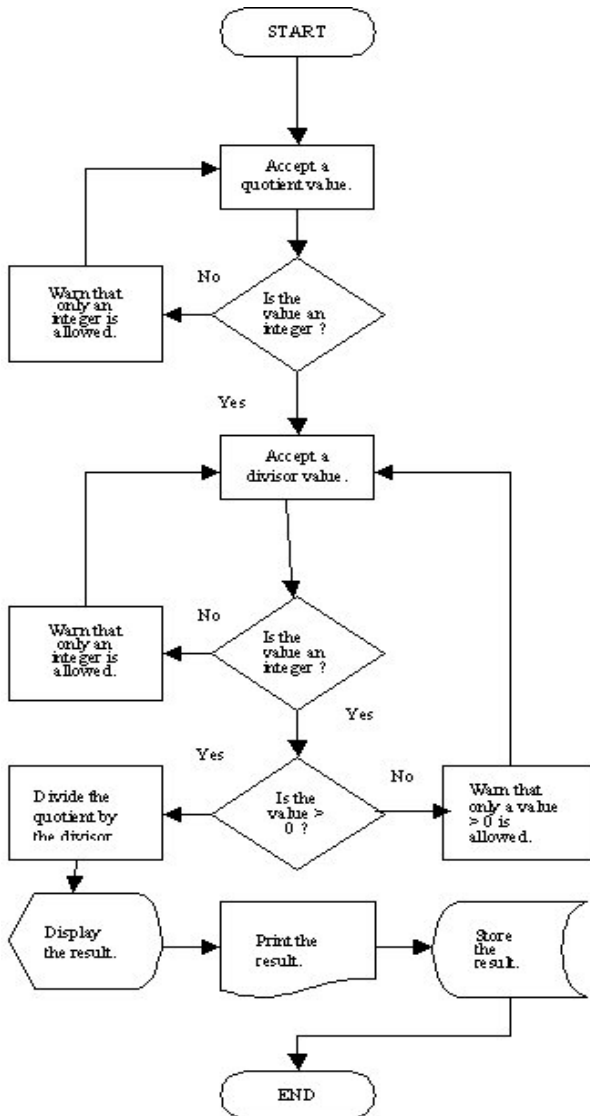


Figure 4. The division tutorial flowchart

6. Conclusion

Flowcharting can give a student a tremendous insight into not only the inner workings of their application but also provide a methodology by which they can explain the

functionality in a clear, concise manner to other people who do not have the same degree of intimate knowledge of it. In this way, the knowledge can be passed on in a structured manner so that others can utilize, enhance and expand the base of information of not only the development of such applications but also of the knowledge domains that can be explored with these precise, effective applications.

References

- American National Standard Institute. (1970). ANSI X3.5-1970. *Flowchart symbols and their usage in information processing*.
- Flowchart. (2008, September 16). In Wikipedia, The Free Encyclopedia. Retrieved September 16, 2008, from <http://en.wikipedia.org/w/index.php?title=Flowchart&oldid=238840385>
- Goldstine, H. (1972). *The Computer from Pascal to Von Neumann*. Princeton, NJ: Princeton University Press.
- ISO. (1985). *Information processing -- Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts* International Organization for Standardization. ISO 5807:1985.
- Supplement to the Massachusetts Mathematics Curriculum Framework; Grades 3, 5, and 7 Grade Level Standards; May 2004, Massachusetts Department of Education.

Author Information

Joseph Frantiska, Jr. is a visiting assistant professor in the computer science department at Fitchburg State College in Fitchburg, Massachusetts. His research interests include hypermedia-based learning environments, instructional design and constructivism. He received his Ed.D. degree with a concentration in educational technology from the University of Massachusetts at Amherst.

331 Wellman Ave.
N. Chelmsford, MA 01863
jfrantiska@aol.com