INCORPORATING STRUCTURED FLOWCHARTING INTO THE INFORMATION SYSTEMS DEVELOPMENT PROCESS

by David Wallace
ACS Department
Illinois State Unv
Normal, Ill 61761

Abstract: The Age of Information and Technology challenges organizations throughout the world to better integrate and implement information technology in order to compete in the world market. Currently, US businesses are experiencing significant problems in information systems development with large software backlogs, a worsening DP/IS personnel shortage, and a lack of reliable methods of producing software. The focus of this paper is on developing more reliable software programs. Effectively integrating software development into the information system development cycle will generate more reliable software by concentrating on the "front-end analysis" stage this cycle. Structured Flowcharting can facilitate the transition from the analysis and design stage of the information system development process (front-end analysis) to the detailed coding and testing stage by identifying and removing errors early in the former stage. This will help alleviate the risk of a major financial loss or massive human inconvenience to the organization. If the bottom line for competition is profit or loss, then the organization can cut its costs and enhance its profitability by building software reliability into its information system development process.

KEYWORDS: MIS, CASE, Systems Development Life Cycle (SDLC), Information Systems, Information Systems Development.

INTRODUCTION

Small businesses, corporations, and organizations throughout the United States are currently involved in an Age of Information and Technology which has affected almost every aspect of our economy (1). One of the most important features of this Age of Information and Technology is the growth in and importance of the world market. October 19, 1987 (the date of the Stock Market Upheaval) emphasized the fact that the United States and other industrial countries throughout the world no longer have only national markets.

The actions of one country's market can have a dramatic impact on other markets in the world. In order for the United States to compete effectively in the world market, US businesses will have to understand the effects that information has on the technological, the economical, the social, and the political structure of their organizations (2).

Organizations that integrate and implement information technology will be the leaders and policy makers within the world markets. Secretary Shultz's address to the Stanford University Alumni Association's First International Conference on March 21, 1986 warned that, for organizations "to seize the opportunities and understand the problems that this new phase of technological transformation will bring, we must grasp both its (information and technology) particulars and its broad outlines."

Currently US businesses are experiencing significant problems in information system developments with large software development backlogs (3), a worsening data processing, information system personnel shortage (4), and a lack of reliable methods of producing software that will meet user requirements (5), (3). These problem areas are major blocks for organizations in their attempts to understand and integrate information technology within their structures. An important step that organizations can take to meet the challenge of understanding and integrating information is to generate methods which can produce more reliable software.

Boehm (6) reviewed studies completed by Jones in 1978, Thayer in 1978, and Boehm in 1980 which all showed that the vast majority of programming errors (65%-75%) are introduced into the early stages of software design. Yet, the removal of these errors from the programs occurs in the later stages of coding and testing. The problem with this result is that the longer it takes to identify and remove errors in the program, the more costly the effort is to the organization. This cost is measured in terms of software failure that could mean major financial loss or massive human inconvenience to the organization. Hence, the most cost effective way to remove errors from the computer program is to eliminate most of them in the design stages, where the costs of correction are generally much lower.

King and Willis (7) conducted a survey of information system directors from a representative sample of the Fortune 500 companies. Based on the results of this survey, the information system directors rated activities in the "front-end" analysis stage (Analysis and Design) of the information system development process as highly important to a successful implementation of computer software programming. Boehm has called the removal of errors in the early stage of software design an indication of software reliability.

A structured design methodology that enhances the identification and removal of errors in the design phase of an information system development process would be an important contribution to software design reliability. Wallace (8) conducted an experiment utilizing two approaches to structured software design in the information system development process. His findings indicated that structured flowcharting helps programming students generate more reliable software. Consequently, the incorporation of structured flowcharting in the information system development process is important to the production of more reliable software programs.

INFORMATION SYSTEM DEVELOPMENT VS. SOFTWARE DEVELOPMENT

Software Development is a subset of the larger process of information system development. Software Development is often called computer software engineering. An information system is "an organized and methodological approach for the collection, storage, and retrieval of data necessary to produce information in a timely manner to satisfy business objectives" (9).

...significant problems in information system developments ... are major blocks for organizations in their attempts to understand and integrate information technology within their structures.

Companies have had information systems set up to provide timely information for hundreds of years. In fact, a company can be thought of as a collection of information systems that support its activities. For example, the United States Army can be broken into five basic information systems:

1. Upper Management,
2. Operations,
3. Maintenance,
4. Research Development, and
5. Separate Disposal.

From these five basic systems, subsystems can be identified that further decompose each of the five. Upper Management can be further exploded into three more subsystems:

1. Direction,
2. Control, and
3. Management (10).

The process of decomposing the information systems into subsystems is known as structural analysis, which is basic to most approaches to systems analysis. Consequently, the information system development process is a methodology that looks at the overall organization in terms of goals and objectives, and that utilizes structural analysis to identify the various systems and subsystems that the organization uses to achieve its goals and objectives.

At specific levels within the information system decomposition (depending on the size and complexity of both the organization and its information systems), systems like the Direction, Control, and Management of the Army Information System will decompose into basic processes or procedures which can be performed either manually or through automation. Software Development can be applied either to existing manual processes or to automated processes, depending on the costs/benefits to the organization. Computer software programming represents less than 15% of the information system development project (9).

From this perspective of information system development, one can acquire a clearer understanding of where software programming fits into the overall development picture. After a cost/benefit analysis has been conducted to determine the appropriate return on investment, the system analyst can verify whether computer programming is warranted for a particular situation. If computer programming is warranted, then, at the appropriate stage in the development process, specifications and actual computer codes can be generated and tested.

Structured flowcharting can facilitate the transition from the analysis and design stage of the information system development process to the detailed coding and testing stage by removing programming errors early in the analysis and design stage. Specifically, data flow diagrams (a tool used in the analysis and design stage) can work in conjunction with structured flowcharts to produce more reliable programs.

DATA FLOW DIAGRAMS

Data flow diagrams are used to model an information system under investigation and to model any recommendations for a new system. The data flow diagram is a communication link between the user and the more technical information system (IS) personnel. With a few basic symbols, the flow of data and the transformation of these data can be represented for any information system (payroll, accounts receivable, etc.). Instead of using pages of written descriptions with terminology that might not be common to the user or the IS personnel, the data flow diagram can summarize this description with a few commonly understood symbols. Recommendations to improve the information system can be made using these same symbols to the satisfaction of all people involved in the development process.

Figure 1 illustrates a data flow diagram for a portion of a payroll system. The key to the benefits of the data flow diagram is its simplicity. This simplicity allows for better understanding and communication between the user and the IS personnel (11). To achieve the benefits of simplicity, a good data flow diagram would incorporate the principles of structured design. Structured design starts with a basic overview of the system which would generally include (five or six) basic processes of the system. Further levels of data flow diagrams are often necessary to explode more detail about the information system under investigation. This idea is often referred to as "Top-Down Design." Figure 2 represents the relationship between the possible different levels in a data flow diagram.

Data Flow Diagrams are supported by additional documentation that further defines basic terminology, data store contents, access to the data stores, and the different processes on the data flow diagram. This documentation is generally identified as the data dictionary. One essential documentation for the data dictionary is the process specification. The data flow diagram labels each process or bubble (See Figure 1) with a strong verb that reflects the transformation of data from input to output.

Examples of good strong verbs for the processing bubbles would be compute, produce, compare, edit, generate, and verify. Further detail is often necessary to explain the use of the verb in the process bubble. When the system analyst uses the term "compute employee's pay", a detailed description should accompany the data

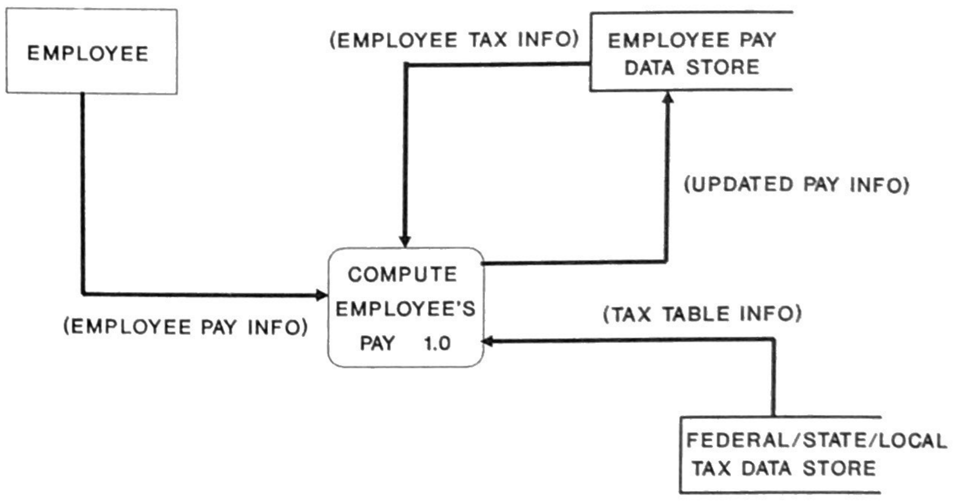

Figure 1. Data Flow Diagram

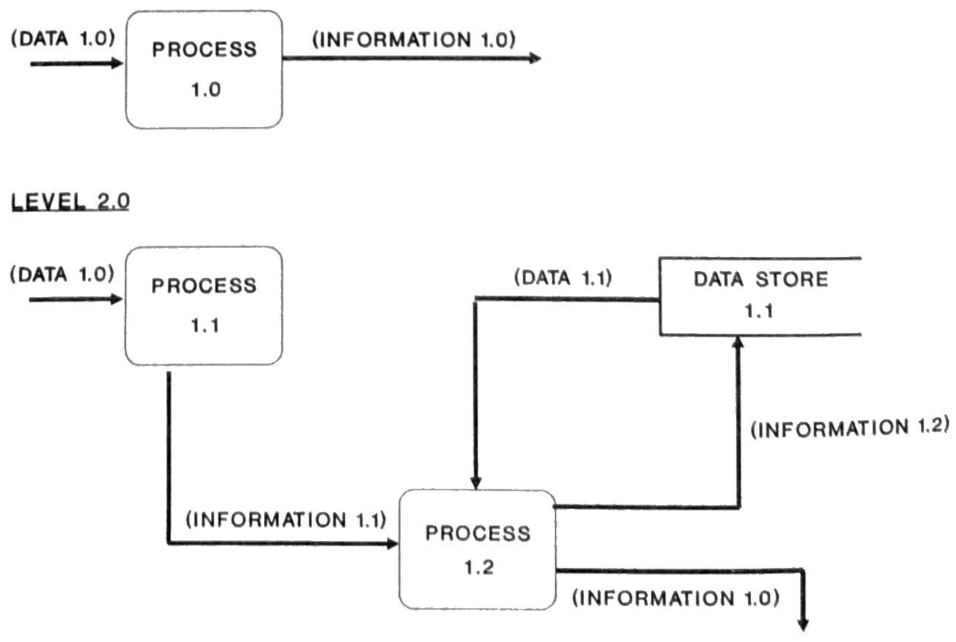

Figure 2. Structured Design with Data Flow Diagrams

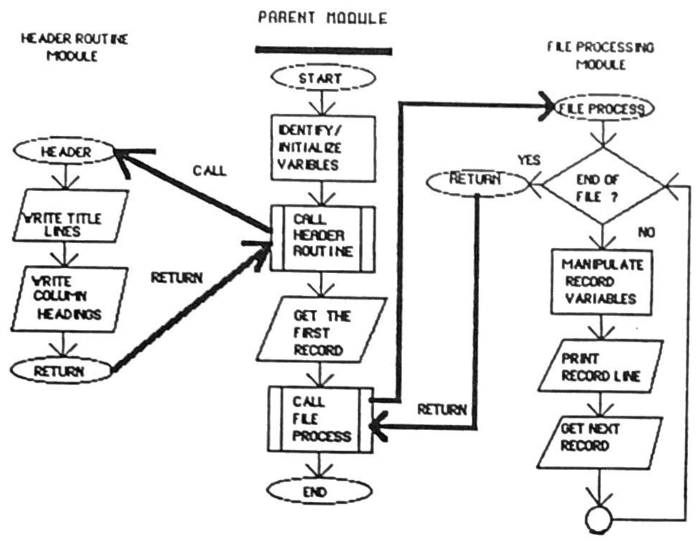

Figure 3. Structured Flowchart

flow diagram, including the appropriate rates to use for Federal, State, and local taxes, any ceilings which have to be taken into account, and any employee exemptions.

The description can be written in structured pseudo-codes (structured English), decision tables, decision trees, or structured flowcharts. Historically, structured pseudo-codes and structured flowcharts have been the most commonly used methodologies for processes requiring software development. By further explaining the processes on the data flow diagrams with structured flowcharts at this point in the analysis and design stage (front- end analysis) of the information system development process, the system analyst can build more reliable software programs (8).

Currently, several information system software manufacturers (NASTEC, Index Technology, etc.) are working on software aids that can generate computer code from flowchart symbols. This research effort will be fully compatible with incorporating structured flowchart symbols into the data flow diagram process specifications since the DFDs are a main tool used in the Computer-Aided Software Engineering (CASE) industry.

STRUCTURED FLOWCHARTING

Structured flowcharting has gone through an evolutionary process over the past twenty years. The early flowcharts were characterized by long stringy symbols reflecting many different logic paths. They were often very confusing and hard to follow. The symbols were also quite restrictive, requiring the programmer to solve the problem in its entirety instead of breaking the problem down into smaller components.

With the widespread use of structured programming in the 1970s and 1980s, the "spaghetti" flowcharts of the past were slowly replaced with other forms of software design tools such as structured pseudo-codes. Yet, a number of programmers/system analysts were convinced that flowcharting symbols still held some advantages as a design tool. The advocates of flowcharting adapted the principles of structured design by using subroutines to break a computer problem into smaller components and by using fewer "go to" statements.

Wallace's (8) research study addressed the use of structured flowcharts as compared to structured pseudo-codes in achieving software reliability development. Students who were taught structured flowcharting as opposed to structured pseudo- coding spent more time designing the solution software design project, less time correcting programming errors and less overall time working on the project.

Figure 3 illustrates a simple flowchart example using structured programming techniques to solve a computer problem. The example is a simple one that focuses on the relationship between the structured process and flowcharting. A more complex example would incorporate the same approach with more levels of decomposition.

During the analysis and design stage (front-end), recommendations are made

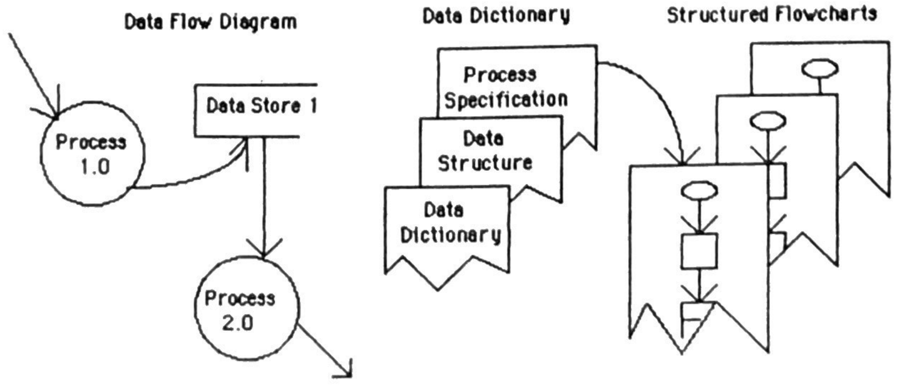

Figure 4. Data Flow Diagrams with Structured Flowcharts

to improve the existing system by adding or deleting data flows, data stores, or processes based on their contribution or distraction to the goals and objectives of the organization. Structured flowcharts can be used to describe any processes involved in generating a computer program. Figure 4 illustrates the relationship between the data flow diagram, the supporting documentation (data dictionary), and the structured flowcharts.

Structured flowcharts should be generated during the Analysis and Design stage of an information system development process. The stages which follow Analysis and Design include detailed coding, implementation, installation, and review of the new system. The structured flowcharts are completed during the "front-end" analysis stage of the information system development process.

CASE tools can be used effectively used to assist the system analyst to model the information system under investigation and to reflect any changes or recommendations to the system. Structured flowcharts can be generated using these CASE tools to further specify those processes involving the generation of computer programming. Currently, information system software manufacturers are developing CASE tools which can go one step further and generate the actual computer code from these structured flowchart symbols. The CASE industry is a relatively new market. Consequently, more research efforts will be needed to enhance and expand these endeavors in order to make them more attractive to organizations. Structured flowcharting is slowly becoming a viable option for system designers.

As an information system goes through the development process, the cost of the project increases significantly. The greater portion of the developmental cost is incurred in the later stages of this development process (7), (5). The equipment is purchased, the software is purchased, specifications are documented and coded, the people are trained, the programs are coded, and the system is tested. The organization has invested much time and money in the project at this stage, and any major errors could be devastating to the organization (5). A problem with software development is that most program errors are injected into the project during the front-end (analysis and design) stage, but are not identified and removed until the detail design and implementation phase, where the cost of the project is significantly higher and potentially even more damaging to the organization (12), (5).

Students who were taught structured flowcharting as opposed to structured pseudo- coding spent more time designing the solution software design project, less time correcting programming errors and less overall time working on the project.

A structured design technique that could identify and remove program errors early in the information system development process would be an important step in building programming reliability. Structured flowcharting identifies and removes programming errors early in the design stage of program development (8). Major errors in design logic can be identified during the analysis and design phase through technical walkthroughs of the data flow diagrams and the process specifications which include the structured flowcharts.

Packaged software products can be evaluated more closely as to the satisfaction of specifications clearly identified on the structured flowcharts that support the data flow diagrams. If in-house programs are more cost efficient, then structured flowcharts are a clear map to the coding process which will occur in the next phase. With the use of structured flowcharts, there should be more continuity between the modeling of the information system under investigation and the actual coding process. Thus, structured flowcharting can be used with the information system development process to build better program reliability. If the bottom line is profit or loss, then the organization can cut its costs and enhance profitability by building program reliability into its information system development process.

CONCLUSION

Structured Flowcharts should be one of the primary tools used to further detail processes identified on the data flow diagrams. As new CASE software products are introduced into the market place which can interpret flowchart symbols into computer codes, structured flowcharts will become more valuable as a design tool which insures software reliability.

REFERENCES

1. Shultz, George. "Age of Information and Technology." Stanford University Alumni Association's First International Conference. March 1986.

2. Greenfield, Gary. "Strategic Planning For a Total Solution." Second International Workshop on Computer-Aided Software Engineering Advanced Working Paper. (1988):21-5.

3. Kahlon, Lakwinderjit. "Realities of Today's Software Development." Second International Workshop on Computer- Aided Software Engineering. (1988):1916- 1919.

4. Baroudi, J. and Ginzberg, M. "Impact of the Technological Environment on Programmer/ Analyst Job Outcomes." Communications of the ACM. 29(6): 546-555.

5. Boehm, B.W. Software Engineering Economics. New Jersey: Prentice-Hall 1981.

6. Boehm, B.W. "Developing Small-Scale Application Software Requirements: Some Experimental Results."

Proceedings, IFIP 8th World Computer Congress. October 1980: 321-326.

7. King, James and Willis G. "The Systems Development Life Cycle: An Empirical Study of the Importance of Development Activities in the SDLC", Twelfth Structured Methods Conference. 1987:179-190.

8. Wallace, David. "Structured Flowcharting Versus Structured Pseudo-Coding in Teaching Problem-Solving Skills in Software Development: A Critical Comparison." DATACON. 1987:119-130.

9. Bryce, M. "Information Systems Engineering and Computer Science Engineering: Do You Know the Difference?" Management Visions. 2(1) 1987.

10. Wallace, David. "Developing a Management Structure Tool Utilizing Automated Modeling Software Package (CASE) for the United States Army." Second International Conference on Computer-Aided Software Engineering Advance Working Papers. 1988:2110- 2116.

11. Powers, M., Adams, D., and Mills, H. Computer Information Systems Development: Analysis and Design. New York: Southwestern Publishing Company. 1984.

12. Zultner, Richard. "Software Quality Engineering: Applying Structured Tools & Techniques." Twelfth Structured Methods Conference - Proceedings. 1987:425- 435.

AUTHOR'S BIOGRAPHY

David Wallace is an associate professor of applied computer science at Illinois State University. He received his doctorate degree from the University of Illinois in 1986, specializing in Management Information Systems. He has presented several articles to both national and international conferences on information systems development and CASE application. His is currently working with the Headquarters, Department of the Army (Washington, D.C.) developing an information system model using CASE tools.

ISSN 1055-3096