STRUCTURED METHODS FOR SAS APPLICATIONS

J. HARVEY TRIMBLE, JR., TRIMBLE DATA SYSTEMS

THE CASE FOR STRUCTURED METHODS

Efforts to standardize the software development life cycle and its associated products have emerged in response to the problems computer professionals were having in delivering correctly functioning applications on time and within budget. To deal with these problems a number of techniques were introduced to bring greater structure and discipline to the computer systems development process. These techniques have been labeled structured methods, and they cover all phases of the system’s life cycle from analysis through design, coding, testing, and into implementation.

The essence of these techniques is that one must try to understand the problem as completely as possible before one tries to solve it. Furthermore, one's understanding of the problem must be documented in such a way that others can read and validate that understanding. Structured techniques emphasize graphic presentations of the problem statement before any solution is considered. They recognize that you must understand what is to be done before you decide how to do it.

Separating the act of solving the problem from the definition of the problem offers the opportunity to consider and reconsider what really needs to be done. It also allows for comparing the results of the application with the problem statement for correctness. The idea that each product of the systems development process should be measured against its requirement to certify correctness is at the heart of the movement to standardize and document the way computer applications are developed.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.

Michael Jackson, in his introduction to PRINCIPLES OF PROGRAM DESIGN, makes the perfect argument for using well defined and disciplined methods in the development of application programs. He writes: "The beginning of wisdom for a programmer is to recognize the difference between getting a program to work and getting it to work right." Although Jackson was writing with high-level programming languages in mind, the point is equally pertinent to applications development using SAS. The ability to develop programs that work right (i.e., do the needed functions correctly) in contrast to those which work (i.e., run without any error messages) requires an understanding and definition of the problem which precedes the act of writing program code.

One feature which makes SAS applications so flexible and maintainable is that data, once defined, flows freely from one process to another keeping its original name and attributes. Applications can thus be developed which are largely independent of the physical attributes of the data they process. The pieces of an application can also be easily rearranged to adjust to new requirements or to include new data transformations as necessary.

Structured methods for applications development fit very well with the features and attributes of the SAS language. Each of these methods uses the input-process-output model for describing computer applications (see Figure 1). This model defines applications in terms of the transformations made to data as they flow from one process to another, and SAS, with its data independent features and powerful data manipulation language facilities, is an excellent data flow processor.
applications. This method produces documentation which serves as a solid basis for contracting with the client regarding what is to be produced. This documentation consists of:

- Data Flow Diagrams;
- Process Descriptions; and
- Data Dictionary. 3/

Data Flow Diagrams are graphic presentations which show the transformations required to change the state of data from input to output. Process Descriptions document the conditions and rules for changing the data. A Data Dictionary identifies the variables contained in each Data Flow and documents each variable's description and physical attributes.

Use of this technique should emphasize the desired outputs first and then derive the needed inputs. Too often we look at problems beginning with assumptions regarding inputs and then focusing on outputs separately. This approach often results in a lost opportunity to understand and simplify the application by failing to focus the analysis on the desired products. An output-driven approach to analysis provides a constant target for function and performance. It also allows the inputs to be restructured to fit the desired result better.

STRUCTURED METHODS FOR DESIGN

Once a mutual understanding is established for what is to be done, the design phase of the application can begin to address the question of how it is to be done. Design is the process of synthesizing the logical requirements expressed in the analysis into physical specifications which can be implemented with a computer language.

Two structured methods have proven to be of substantial benefit in developing SAS application designs. First, Yourdon Physical Data Flow Diagrams do a good job of showing the combination of SAS Procedures and Data Steps needed to meet the requirement. These Data Flow Diagrams have the same composition rules as those done for analysis, but they are used to determine the physical way the problem is to be solved.

Second, program design methodologies driven by analysis of an application's data structure, such as those promoted by Ken Orr 4/ and Michael A. Jackson are very useful in determining Data Step design. These methodologies require programs to be structured in direct correspondence with the structure of the data being processed. Both the Jackson and Orr methods use graphic techniques to express program (Data Step) design. These graphics are simple to produce and are very effective in developing and communicating the structure of programs.

Michael Jackson's program design procedure consists of the following steps:

- Define the data structures (both input and output);
- Create the program structure which models the data structures;
- Express the program functions as tasks to be executed and
- Fit the program functions to the program structure. 5 /

It is important to note that Jackson's approach yields one and only one program design for an application - a claim that no other program design methodology can make. It is a non-inspirational design methodology which can be validated by those inside and outside the development team.

STRUCTURED METHODS OF CODING

Once the questions of WHAT and HOW have been answered the remaining tasks are to code and test the application. Structured coding techniques for SAS Data Steps follow the same approach used for procedural languages, including:

- Use of a few simple program constructs; and
- Use of simple composition rules.

Consistent with established structured programming concepts and techniques, the constructs which have proven to yield the most straightforward program code are: 6/ 

- Sequence (i.e., one statement followed by another)
- Selection (i.e., IF-THEN-ELSE)
- Repetition (i.e., DO-WHILE)

These constructs are also the same as those used to transform logical analysis into physical program design. They minimize the distraction of jumping from place to place in the program to another by severely limiting use of GOTO statements.
Since program readability is an important short and long term objective, following these straightforward techniques reinforces that goal. Readable programs reduce development time because they are easier to understand and test. They also reduce maintenance costs because someone besides the original author can read, understand, and change the program.

STRUCTURED TESTING

Structured testing encompasses a number of methods to help assure that the program produces correct results. Above all it requires construction of a test plan which details how the application is to be tested and what it will be tested for. Test data which will produce the desired test results must be developed along with the test plan. This test plan and its associated test data should be saved and used to certify that future application changes do not adversely affect application correctness.

SAS applications are relatively easy to test since procedures to print or tabulate the inputs and outputs to a process can be added at any time. Frequency distributions of the inputs and outputs of a Data Step or procedure are excellent tools to test application correctness.

The key point about testing is never trust the results of a program. We cannot accept on faith the results of programs which appear to work. All programs must be rigorously tested before we can accept their results as correct. But even after acceptance, we have to be alert to the insidious cases which cause programs to produce the wrong results, thinking all the while that they are correct. Program logic errors occur due to combinations of circumstances, and there are too many combinations of conditions in a program to test exhaustively.

STRUCTURED WALKTHROUGHS

Structured walkthroughs consist of the formal presentation of one's work to others. Walkthroughs work to eliminate the notion that application development is a solo endeavor. They are conducted during each phase of the development, and are a vital part of the methodology. They help insure that the appropriate documents are correctly and legibly prepared, and offer an opportunity to review and correct each product as it is developed.

They serve additionally as a quality assurance check, and they offer an excellent opportunity to develop the skills of all participants. Moreover, they work to yield homogeneous products from the development team and, since everyone's work is reviewed, standards naturally evolve and are self enforcing.

IMPLEMENTATION OF STRUCTURED METHODS

Introducing structured methods to an organization requires a concerted effort which should have the following elements:

- Awareness;
- Education; and
- Review and Evaluation.

The first step in introducing structured methods to an organization is to develop a management awareness that new system development techniques are desirable. It is important to establish a consensus that improvements are possible. Second, the education of key staff in the theory and application of this methodology is necessary. This education should not be limited to computer professionals alone since the ultimate success of using these methods depends heavily on the cooperation and involvement of end user specialists and organization management. Finally, periodic review and assessment of how well this methodology works in the organization is vital to customizing it to fit individual projects and helping to institutionalize the approach.

CONCLUSIONS

Applying structured methods offers the opportunity to substantially improve the productivity of those developing SAS applications. A set of well developed and thoroughly tested methods exists which can substantially improve everyone's understanding of what the problem is and how it is going to be solved. However, the most important result from applying these methods is not increased productivity or better communications but rather more reliable and correct solutions to the organization's problems.

FOOTNOTES


5. Jackson.


The author may be contacted by calling 703-759-3053 or writing:

J. Harvey Trimble, Jr.
TRIMBLE DATA SYSTEMS
10706 Wynkoop Drive
Great Falls, Va 22066