DEVELOPING A SAS TRAINING COURSE
Jean Ussery, SAS Institute Inc.

Abstract

If you are assigned the task of developing a SAS training course, where do you begin? How can you decide what topics to cover and how to organize them into a logical sequence? This paper discusses a structured technique used at SAS Institute to analyze the skills students need to learn and the development of a course based on that analysis. Materials developed may be in a variety of forms—overhead transparencies, 35mm slides, videotapes, or text for a workbook or computer-based instruction. Since many people are concerned with developing an introductory SAS course, this paper uses topics in such a course as examples. The techniques, however, can be applied to the development of any course, at any level of experience.

Why Have a Method?

Why is a method needed for course development? Designing a course is like writing a computer program. Programmers that follow structured design techniques usually complete assignments faster, since repetitive components are written only once. Structured programming methodology provides a known starting point: instead of floundering in a sea of ideas, the programmer uses a rational approach to problem solving.

Course developers need a similar technique. Students learn best when material is presented in a logical fashion and in small, complete units. Common course design errors include planning too much material for too little time, including topics that are not related to the students' learning goals, and presenting topics at the wrong time. Like structured programming, structured course development enhances organization and efficiency.

For example, an introductory course might include a discussion of three forms of input: column, list, and formatted—and the use of the trailing @ (at-sign). Without a course design methodology, these topics might be lumped into one discussion. With a design methodology, the topics would appear at different points in the course. Column and list input may appear early, while informats may be introduced later in a discussion about reading non-standard data files or using SAS date and time values. The trailing @ might be discussed with special input problems or eliminated altogether if it does not support the course objectives.

A course development plan provides a means for selecting topics and organizing them into a logical sequence. Courses can also be modularized easily. This allows students to select topics and increases the flexibility of your training program. Unnecessary topics are excluded from the course. By following a design methodology, you can develop courses that achieve the training results you want.

The Course Development Plan

The plan consists of five phases.

1. Investigating needs and available resources
2. Designing course content and flow
3. Writing the course outline and materials
4. Testing and revising presentation and materials
5. Finalizing course materials and content

Investigation

The investigation phase of course development includes two stages: needs assessment and research. In your needs assessment, you define what needs the course is expected to fulfill, for whom, and how the course relates to other courses in your training program. In the research stage, you examine available literature and similar courses and discuss ideas with subject experts and students.

Needs Assessment: To assess need and expected content, talk to the people requesting the new course. They should explain why the course is necessary and define the intended audience. Find out how potential students are currently learning the material. Learn the specific new skills students should have after taking the course. What backgrounds will students bring into the course? (Consider computer skills as well as SAS programming abilities.) If you have a training program, determine how the new course relates to other courses in the program. Ask for names of potential students and users experienced in the subject areas.

Research: Examine printed materials related to course topics and investigate other courses in the subject area. If such courses exist, you may want to attend a class or get copies of the course materials. You may even discover an existing course that will meet your needs. For teaching SAS programming, remember to look at past issues of SUGI Proceedings, since many papers address SAS training.

As part of your research, interview content experts—those users within your organization who have mastered the skills to be taught. They will have tips and sample programs you can use and can recommend documentation to support the course.
Also interview potential students. Find out their backgrounds and their goals for the class. What specific tasks do they want to be able to perform after the course? If the method of course delivery (instructor-led, video, etc.) is up to you, ask the students which method they prefer. What is their time frame for learning the course topics? How will they use the skills taught in the course to perform their jobs?

Design

After investigation, you enter the design phase of course development. This phase includes defining pre- and post-course skills yardsticks and performing a task analysis. The two skills yardsticks are required during task analysis. As you analyze potential topics for the course, the yardsticks help determine which ones should be included. From the schematic of the course produced through task analysis, you can outline the course topics, divide them into teaching units, and develop course materials.

Skills Yardsticks: From your needs assessment and research, define your audience. Describe the computer experience and SAS programming skills the students are expected to bring into the course. Define the new skills students should have attained upon completion of the course. Be specific in defining your yardsticks. Note the concepts as well as the SAS statements and system commands students should be able to use.

If you are designing an introductory course for SAS users, you might establish the following criteria as your pre-course skills yardstick:

This course is designed for the person who can log on and off the computer system, use the terminal, and access the editing (text) editor. The student should be able to use the editor to enter information into a file, delete lines, insert new lines, change text, scroll through the file, locate text strings, and write the file to disk (or other storage medium). The student should be able to issue system commands to create computer files as required, access external files needed by a program, produce printed copies of the contents of a file, and delete files from storage that are no longer needed. The student should know how to name new files according to installation standards.

At the conclusion of the course, the students should have learned a new set of skills. In an introductory course, you might define the following post-course skills:

At the conclusion of this course, the student will be able write and execute a SAS program to access an external data file stored in standard-numeric column-oriented form or list form (data values separated by at least one blank). The student will be able to write a DATA step to read the data file and create a temporary SAS data set, use assignment statements to create new variables or modify the values of existing variables in the data set, control conditional processing with IF-THEN/ELSE statements, and control observations written to the data set with the subsetting IF and DELETE statements. The student will be able to write a PROC step to invoke a SAS procedure; know what tasks can be performed by the PRINT, SORT, and MEANS procedures; and use ID, VAR, and TITLE statements to modify the appearance of the reports produced by these procedures. The student will be able to control subgroup processing with PROC PRINT and MEANS, and use these PROC MEANS options: MLAN, SUM, MIN, and MAX.

The skills yardsticks are extremely detailed, to ensure that topics unrelated to the specific objectives are kept out of the course design. The skills yardsticks become the prerequisites and global objectives for the course. Once you have defined these yardsticks, begin analyzing the tasks the course should teach.

Task Analysis: The required knowledge between the pre- and post-course skills define what is to be taught in the course. Begin your task analysis by listing all tasks related to the course topics. A task is a complete unit of work, such as writing a DATA step to read a column-oriented external data file and create a SAS data set. Consult the course requestors for the tasks they think the students should be able to perform at the conclusion of the course.

Next, analyze each task by comparing it to the pre-course yardstick. If the task is defined within the pre-course skills, it should not be included in your course. If you are designing a review section for the course, you may include it there. If you are designing a series of SAS courses, you may note that the topic needs to be included in the course prior to the one being developed. If you plan a pre-test for the course, you may want to include some test items related to this topic.

Tasks are also compared to the post-course skills yardstick. If the topic is beyond the level indicated by the yardstick, it is not included in the course. (Again, you may want to note this topic for a future course.)

If a topic appears to belong in the course, apply one more test: is the topic "need to know" or "nice to know?" Nice-to-know topics have no place in your course. Remember, students assimilate a lot of new information during a course. Nice-to-know topics can confuse the learners. For students who want to pursue such topics, you may include a list of suggested readings in technical manuals or some optional programming projects in the course materials. A topic could be nice-to-know in one course but need-to-know in another.
For example, based on the post-course skills yardstick, the use of the trailing @ as an input technique should not be included in the sample course. The trailing @ topic does nothing in the course to progress the students from their pre-course skills to meeting the post-course objectives. Therefore, it is not an appropriate topic for this course.

Next in the analysis phase is defining the details of the tasks selected. Imagine looking over someone’s shoulder as they accomplish the task. List each step necessary to perform the task. Task detailing may reveal gaps in the skills yardsticks or indicate new tasks that should be included in the course.

Task analysis provides the backbone of the course. Most developers find the analysis phase the most difficult. At each step of the process, you may have to change your criteria and re-analyze. Task detailing may indicate new topics that change the skills yardsticks. When you are satisfied that the appropriate tasks are included and detailed, organize them into a logical sequence. From this point, the course can be outlined and instructional materials written.

Writing

From the schematic of the course produced by task analysis, you can develop a course outline and produce materials to support the presentation.

Course outline: First, outline the course topics and divide them into instructional units. State the objectives for each unit to ensure that all global objectives of the course are met. Determine appropriate points for learning reinforcement with exercises and computer workshops. Keep each unit small but complete.

Course Materials: For each instructional unit, design a series of handouts, graphic displays, glossaries, and so on, to support the course presentation and the learning process. Be sure to use clear, concise instructions in workbooks or self-study manuals. If you are designing slides or transparencies, use key words for a clean, uncluttered look. Students like to receive a copy of the visual displays as an aid to their note taking. Use a consistent format for the materials.

Select two or three data files to use in sample problems and workshops. Using only one data file gets boring; too many different files forces the students to concentrate on the data rather than learning new SAS programming techniques. Use sample data files like ones the students will encounter in their jobs. You may even have the students code and execute SAS programs they need on the job as part of the course.

Testing and Revision

Course design and materials are tested in two ways: a review by content experts and a sample class. Based on feedback from the reviewers, you can improve the materials or design.

Review by content experts: Before offering your new course to students, have your materials reviewed by others who know the subject matter. Experts can find errors in sample problems and inconsistencies in terminology, and can recommend areas that need modification.

Sample class: Hold a sample class to test the course design, timing, and flow before going into full production. Select a representative group of students and give them an opportunity to comment on the course design and materials. Keep copies of the materials in the classroom for the students to write their comments and to note your own observations.

Ask for feedback on the timing of the course and points of confusion. To fit a specified amount of time, you may need to adjust some of the topics or exercises. Note the points in the course where many students were confused; this may indicate areas where reinforcement of course materials is needed. Beware of errors in your handouts! You certainly do not want to give the students sample SAS code containing errors or text with grammatical errors. Pay particular attention to written instructions of computer exercises. (For example, in an introductory course, some students press the ENTER key only when directed to do so.)

If you plan to convert your design to a self-directed medium (such as video or computer-based courses or a primer), it is most important to test the design. You may hold a sample instructor-led class or work up a prototype of the course for testing. You do not want to expend too many resources on the prototypes; valuable time and money may be wasted if you make extensive revisions to the design.

Finalization

You could go on making revisions to the course indefinitely; many course developers seem to be perfectionists, and therefore never finish revising the course. At some point, freeze the course design and content, print multiple copies of the course materials, and offer a standard course to all students. If the course will be taught by others, hold a class for them on how to deliver the new course or prepare an instructor’s guide.

Conclusion

Developing a course is like designing a computer program. With the aid of some structured techniques and rules of thumb, you can design courses quickly to meet the needs of the students.

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.