Program Development

By Karen Overfield

Most train-the-trainer courses and textbooks emphasize a structured, linear approach to program development. But the world isn't always a structured, linear place. Here's an effective approach for people who look at program development as a system rather than a series of ordered steps.

Pick up a training magazine, journal, or book, and you'll probably see an article, chapter, or section on instructional systems design. Attend a training conference, and you're likely to find at least one session about training program development. If you prefer, go to a workshop, take a class, or enroll in a degree related to curriculum design. Most will emphasize a structured, linear approach to training program development.

I have a confession to make. I don't think in a linear manner. I never have, although I've tried. Maybe you don't think that way either.

While working on one phase of a training program, I used to find myself thinking of something that belonged in another phase. I'd usually write it down, stick it in a file folder, tuck it away in a safe place, and return to what I was doing. But my mind would keep drifting back to that forbidden thought or idea. I felt like a failure—a traitor to my profession. I just couldn't force myself to work in a linear manner.

Most experts agree on several set phases for program development: needs assessment, analysis, design, development, implementation, and evaluation. That's all fine and good, but what do you do with administrative functions and details? Where do the day-to-day operating procedures fit in? When do you schedule people for classes, pay invoices, and order supplies?

Overfield supervises the training programs in the information systems department of Mobay Corporation, Mobay Road, Pittsburgh, PA 15205-9741.

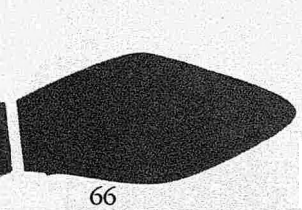
Those types of duties continue throughout the life of a training program. They don't end with implementation; in fact, they really only begin there. Training programs represent ongoing, cyclical functions. With a linear approach, you must start at the beginning, but often that approach is not practical.

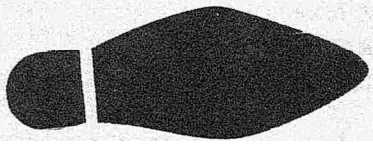
A linear approach also takes time. It requires you to determine its critical path by adding up the estimated length of time for each phase. Quite frankly, I don't have the time to do things that way.

Still, I'm one of those practitioners who ends up married to the program I'm working on. I can't just walk away from it after implementation. I need to deal with continuing program administration, maintenance, and revisions, and I need to work on two or more phases at a time. A method for cutting the length of a project's critical path could save me a lot of time and unnecessary headaches. Perhaps it could do the same for you.

A synergistic approach

Last year, my company's management charged me with developing a training program for the application-systems group of our information-systems department. I knew I wanted to take a practical, workable, and realistic approach in developing the program. But with limited time and resources, I faced a dilemma: should I do things by the book—that is, use a linear, structured approach—or should I risk doing it my way?





for the Real World

I opted for the riskier route: to look at my department and its training needs from a systems point of view. To me, any training program represents a system made up of several, separate subsystems. Those subsystems are synergistic in nature—that is, the whole becomes greater than the sum of its parts. So I created what I call a Systems Program Development (SPD) model.

SPD represents a dynamic organizational development intervention. As the program developer, you continually look at symptoms, problems, and needs. Then you use that information to design a program to fit your organization. SPD differs with each organization, the training problem, and the person using it. You must use every SPD subsystem, but the extent to which you use each subsystem and the order of their use can vary.

With SPD, each subsystem contains identifiable inputs, processes, and outcomes. Those subsystems, although separate, affect each other and the program as a whole. For my training program, I identified the following subsystems:

- skills identification;
- needs identification;
- analysis;
- administration;
- program design and development;
- implementation;
- program evaluation.

The analysis subsystem was central to my program. In SPD, analysis represents an ongoing process that directly relates to each subsystem. You must do an analysis for each subsystem, not just one separate phase.

The unstructured approach of SPD involves a good deal of risk taking. You don't have a step-by-step, linear methodology to follow. Instead, you develop individual training programs to address the changing requirements of the people in your organization.

Even though the training program I worked on involved an application-systems group, SPD is not confined to computers or technical areas. You can apply the model to any training program. The following is not a step-by-step, cookbook-style recipe for successfully developing a training program applying the SPD model, but a practitioner's view of the model I used and how I used it.

Skills identification

The first subsystem on which we focused was skills identification. The inputs of this subsystem are the people, their positions, and the tasks they do as part of their jobs. The outcomes are the skills needed for the job, the skill levels required, and the audience that may need training.

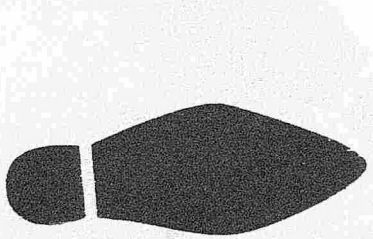
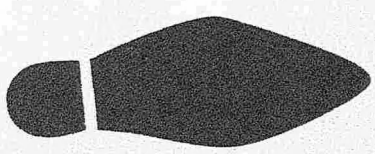
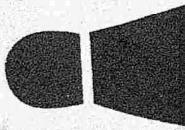
As you might guess, success in this subsystem is not a trivial task. It takes time as well as two other critical elements: management commitment and people involvement. Often you may feel tempted to skip the skills identification step. I tend to want to get right in and do, rather than sit back, analyze, and plan. From my experience, bosses are like that, too. They like results; they don't like to hear theories or get the impression that you're making a career of doing a skill analysis.

But if you try to ignore skills identification, you're headed for a collision. You need to stop and identify the skills needed for the job, look at the skill-level requirements, and then see how they fit together.

Because our department is large, diversified, and divided into work groups, we chose to implement the training program in stages, taking one work group at a time. SPD lent itself nicely to that approach, as it allowed us to move freely from one subsystem to another as well as do things out of sequential order.

The first group served as a pilot. We developed techniques, refined them, and applied them with the next work group. We chose the application-systems group to pilot the approach for several reasons:

- management had already established a career path;
- job descriptions existed for the various positions;
- the former training program could serve as a reference point;
- the people wanted training;
- management was committed to providing the training. In fact, management named one of the managers to work closely with me and to act as a liaison between management, staff, and me.

Position/skills matrix

In the application-systems group, the career ladder consisted of different levels and career alternatives. For instance, when people reached the level of Programmer Analyst III, they had to make a career choice. They could follow either a technical route leading to Senior Programmer Analyst or a management route leading to Senior Systems Specialist. To that point, people in the various positions did basically the same type of work. Differences involved skill levels, internal and external relationships, accountability, and degrees of supervision.

A primary key to the success of our program lay in the involvement of the people. We got them involved from the beginning. All training programs need to meet the requirements of the people for whom they are designed. To make sure they do, you must get the people to define the skills needed for their jobs. To do that in our program, we formed a task force.

Management selected three people from each position to serve on the task force. Before the task force met, I drafted a skills inventory based on the job descriptions, information I found through research, examples from other organizations, my own experience, and the previous training program.

The skills inventory served solely as a starting point. The people on the task force identified the actual skills and defined their competency-level requirements. To identify skills and competency levels, we set up a series of two-hour meetings on company time with members of the task force, the manager, and me. We met separately with members of the task force from each position.

Before the first group met, I distributed copies of the initial skills inventory to them. At the meeting, people in the group contributed their ideas, suggestions, and knowledge of the work to revise the profile and define the competency levels they felt were required for the "ideal" person in their positions. Members of the task force showed no hesitancy in changing, adding, or deleting skills. We were working on something that was very important and close to them.

All training programs need to meet the requirements of the people for whom they are designed. To make sure they do, you must get the people to define the skills needed for their jobs

We used these designators for skill competency requirements:

- N for not needed (the skill is not required for the job);
- A for aware (employee should know a little about the skill);
- C for confident (employee should be able to use the skill on the job, solve day-to-day problems, and handle routine troubleshooting);
- P for proficient, or expert (employee should be able to perform at an advanced level, identify potential problems, provide guidance, and answer questions).

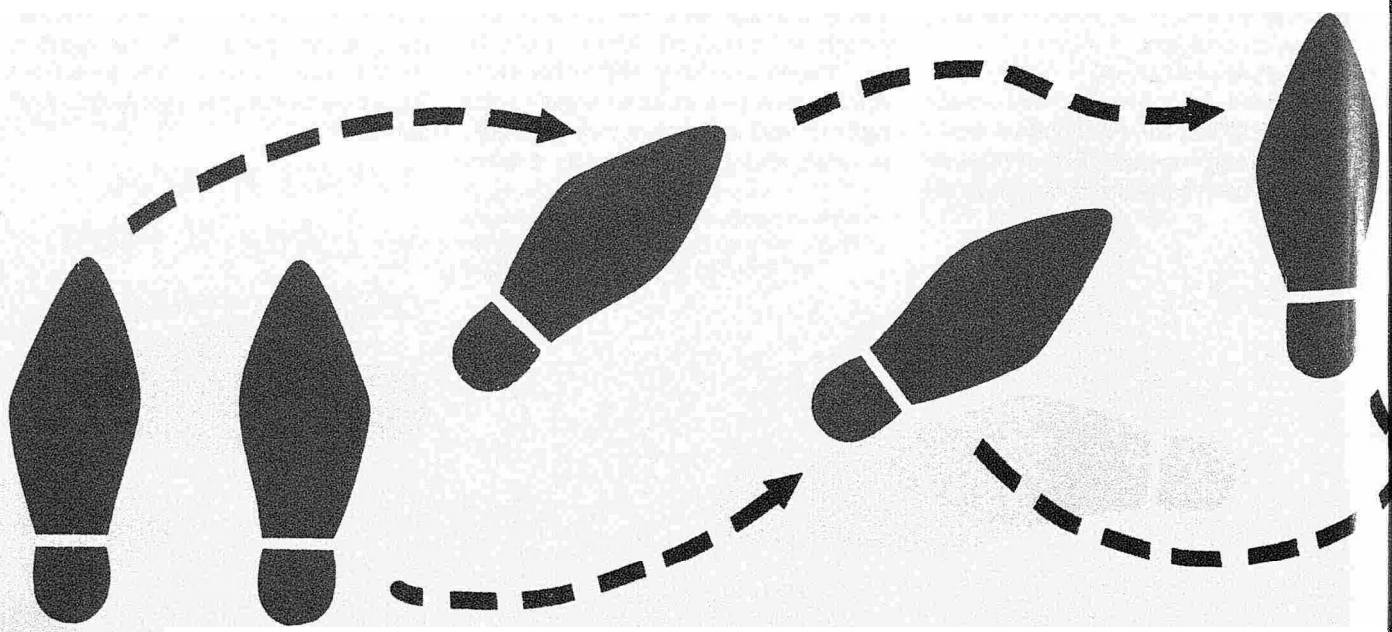
At the end of each meeting, we asked the task group to give input and to rate the competency levels of the positions both above and below theirs. That served as a starting point for discussion with the next group. I revised the profile and distributed the revised copy to the people in the next group to review before our meeting. We continued with the procedure until we had met with people in each position.

After meeting with each group separately, we held a joint meeting with one representative from each position. At that meeting, we finalized the position/skills profile and identified where training, if needed, should begin.

Once we felt comfortable with the position/skills matrix, we presented it to management for approval. Figure 1 shows a section of the matrix. We agreed to handle individual job differences on an exception basis later, during the individual training-needs interviews.

Next, I used the position/skills matrix to develop a training curriculum for the "rookies"—the new hires. When we began the process, several job openings existed in the work group. Management wanted a training program in place when the new group came on board.

Using the position/skills matrix, I researched existing courses that met the specified objectives. That activity falls under the category of design. If I had followed a linear approach, I would have had to wait until I got to that phase, and management would



have had to wait for a training program for new hires.

With SPD, I moved to the design, analysis, implementation, and administrative subsystems. I identified courses, costs, and time commitments for training. I developed a curriculum, prepared cost figures, wrote up purchase orders, and made my recommendations to management. Management signed off on the curriculum, I ordered off-the-shelf courses, developed administrative procedures, and put the curriculum in place. Management began to recruit. Using SPD, I was able to combine several subsystems, greatly reducing the time of putting part of the program in place. Management got what it wanted, and I was able to go on to needs identification.

Needs identification

The needs-identification subsystem uses the inputs of skills inventory, skill-level requirements, and the prospective audience to develop a process to identify the outcomes of the training needs, the trainees, and the training priorities.

Employee training differs from employee education, or development. Training, which carries a sense of urgency, prepares people to do their current jobs. Training should stem from gaps in knowledge or performance. Development, on the other hand, prepares people for their next jobs. It provides a source for personal or professional growth and education. Management wanted our training program to address both. Once a person's training needs were met, we then could address development needs.

Training needs differ from employee to employee. Not everyone doing the same job needs the same training. Some people have the skills they need, some get them on the job, and others need formalized training.

Training should provide people with the skills they lack to do their jobs. Through needs identification, you identify performance gaps and training needs. The goal is to identify the skills people need by comparing their current skill levels to some standard. A direct relationship should exist among job knowledge, needs-specific training, and on-the-job performance.

Employee training differs from employee development. Training prepares people to do their current jobs. Development, on the other hand, prepares people for their next jobs

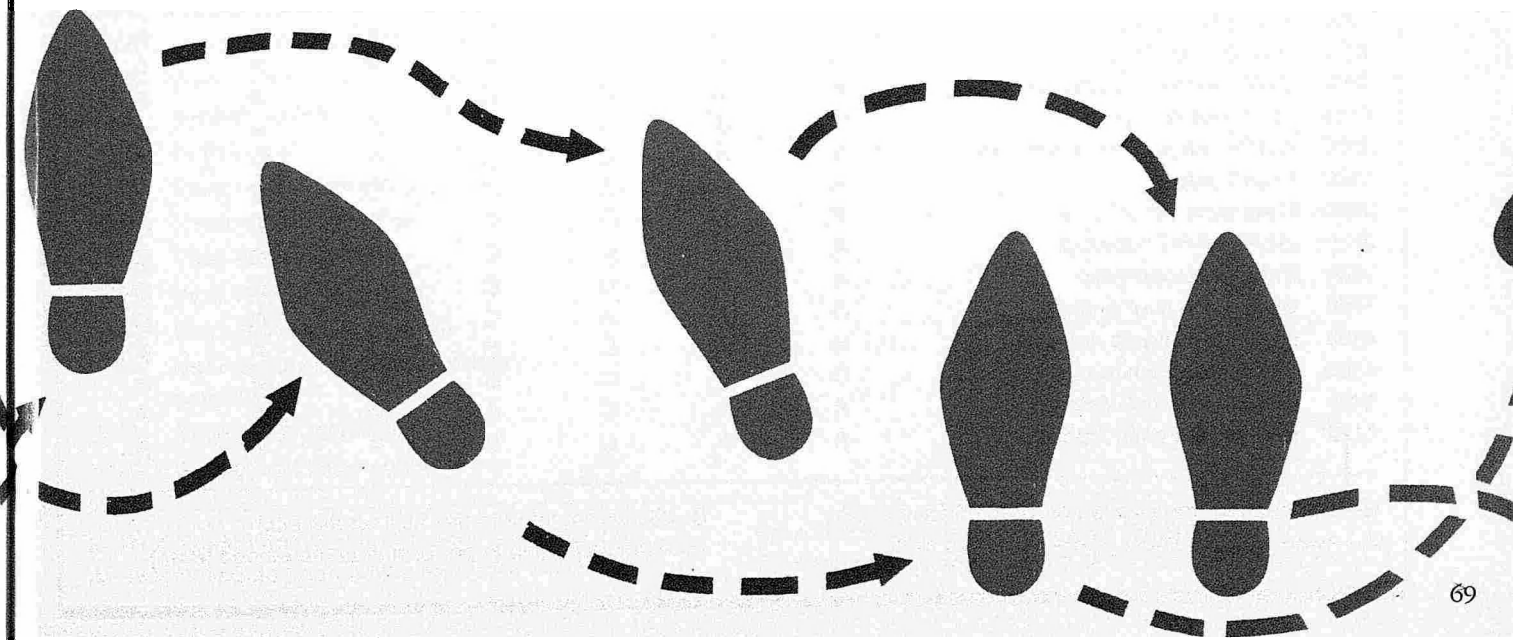
In our case, we used two-step, individual interviews to identify training needs. I conducted the initial interviews to identify needs. Individual managers conducted the second round of interviews to identify the commitment to the trainee's training plan.

To conduct the needs identification, I gathered data from the position/skills matrix and created forms that listed the skills and competency levels for each position. I included space for the trainees to indicate their current competency levels, whether they felt they needed training, and their priority for the training, as well as space for me to fill in the time frame. Figure 2 shows a section of the needs-identification form for the Programmer Analyst I position.

Before beginning the interviews, management and I met with the application-systems group members to orient them to our approach for developing their training program. At the meeting, we told them about the process we had used to identify skills and competency levels, the approach that would be used for the needs interviews, and the guidelines that had been established for training in their group. We answered their questions and gave them time frames for completing the process.

I scheduled an individual training-needs interview for each person in the application-systems group. Before each interview, I sent the person a copy of the position/skills matrix, the appropriate needs-analysis form for his or her position, and definitions of the skill-level identifiers.

Before the interview, each person was to fill in the column of the form that identified his or her current competency levels for each skill listed. During the one-hour interview, we discussed individual and job differences in required skills, training needs and whether they were being met, and development needs. As the interviews progressed, I saw patterns in the training needs among the trainees and began to research alternatives for meeting them.



After the initial interviews, I prepared a learning contract for each trainee and gave it to his or her manager. During the second interview, the managers went over the contracts with the trainee, and made any necessary changes. Both people signed the contract. We used that procedure because

- I represented an unbiased, neutral, third party. Trainees rated their competency levels and identified their skills gaps. I accepted them without judgment. Any differences between the individuals' evaluations of their training needs and their managers' evaluations would be handled during their discussions with the managers.
- People trusted me. No past history existed around me or the SPD approach. I had gained credibility through working with the task group.
- I needed to build a bond with people in the area. This process was a good way for trainees to get to know me and feel comfortable coming to me to express training needs or concerns.

The SPD allowed me to work on four separate subsystems during the interviews: design, administration, analysis, and needs identification. That flexibility enabled me to cut program-development time considerably, as well as facilitate it. Because I had training needs and objectives fresh in my mind, if I wanted to know if an existing course could meet or was meeting a training requirement, I could mention it during the interview.

During the pilot needs interviews, I addressed training primarily from a skills standpoint, because I hadn't yet researched any options and we didn't have a program in place. Since implementing the program, I give people training options and a chance to give input during the needs interviews. Giving them a choice about the training that directly affects them is empowering and helps them gain a sense of control over the training.

Analysis and program design and development

The analysis and program design and development subsystems also represent continuous, ongoing processes. They are not isolated stages, but integral parts of a synergistic whole.

In analysis, specific inputs and outcomes differ, although the process remains much the same. As the program developer, you evaluate the alternatives based on certain criteria, make a recommendation, and identify what you will do and how you will measure its success. Various tools exist to help you in analysis—from sophisticated statistical, quantitative techniques to qualitative techniques. What you use depends on your work style, your individual preferences, and the situation.

In program design, the training needs and priorities you have determined become training events and a curriculum; individual employees become training participants. It is helpful to keep in mind that research emphasizes the importance of adult-learning concepts and principles in training. Also remember that people differ in learning styles, rates, retention, and frames of reference. Both affect program design and development. I defined these goals for our training program:

- to use a variety of training methods and media;
- to apply principles of adult-learning theory;
- to encourage participation and involvement;
- to provide opportunity for application, reinforcement, and refreshers.

Implementation, administration, and evaluation

Implementation should be an exciting experience. If you don't make a big thing out of this subsystem, no one else will. To build enthusiasm for our program, we developed several learning aids:

- A course catalog added credibility and professionalism to the program. People in the department used it to find information about specific courses and the types of training available.

Figure 1—Position/profile skills matrix

Code	Skill description	PRGMR ANALYST	PRGMR ANALYST I	PRGMR ANALYST II	PRGMR ANALYST III	SENIOR PRGMR ANALYST	SYSTMS SPCLST	SENIOR SYSTMS SPCLST
1100	Manages projects	N	A	C	C	P	P	P
1200	Develops systems	N	A	C	P	P	P	P
1300	Uses problem-solving process	A	C	C	C	P	P	P
2100	Plans use of time	N	A	C	C	P	P	P
2200	Applies ways to reduce stress	N	A	C	C	P	P	P
3100	Uses Lotus	A	C	C	C	C	C	C
3200	Uses word processing	A	C	C	C	C	C	C
3500	Identifies PC concepts	A	C	C	C	C	C	C
3600	Uses electronic mail	A	C	C	C	C	C	C
3700	Uses voice-mail system	A	A	A	C	C	C	C
4100	Does cost/benefit analysis	N	A	C	C	P	P	P
4200	Uses performance-mgt. process	C	C	C	C	P	P	P
5110	Recognizes terminology	A	A	C	C	P	P	P
5120	Recognizes basic concepts	A	A	C	C	P	P	P

N—Not needed (the skill is not required for the job.)
A—Aware (should know a little about the skill.)
C—Confident (able to use skill on the job.)
P—Proficient (able to perform at an advanced level.)

■ Student learning contracts illustrated management's commitment to training, gave the trainees an idea of the training they would receive, and served as documentation for us in scheduling courses and planning training events.
■ Posters and flyers were created with a PC graphics package to announce training events.

The administration subsystem involves collecting statistics that relate to program standards, delivering the training events, and maintaining day-to-day working procedures.

Working procedures include such activities as how to notify people of training options, who enrolls people, who gets billed, and who handles payments. Such clerical activities consume a lot of time, require accuracy, and are repetitive. But if you don't pay close attention to them, your program won't run smoothly. They touch everyone involved in the program and can make or break the program because of their visibility. Test, refine, and retest your procedures before using them.

Documenting working procedures is also vital, as it forms the basis for evaluating the program. It is best to document things as they occur, no matter how inconvenient that might be. If you don't have something to refer back to later, you're "up a creek."

The administration subsystem lives on after the development ends. If you don't spend adequate time refining it, you'll feel like you're married to it. Try to identify ways to work smarter. Streamline administrative procedures and eliminate duplicate and rework efforts by making sure you know the purpose behind what you do. Do away with anything that doesn't directly tie back to program goals and objectives. If you don't know why you're doing something, don't do it.

Like the other subsystems, the final subsystem, program evaluation, must be an ongoing part of the process. In our department we use evaluation forms, informal interviews, and follow-up phone calls to get input on the success of our program. We also send out quarterly reports to managers.

The real world

Theories sound nice on paper, but taking a theory right out of a book or an article doesn't work. When you're faced with time constraints, multiple projects, budget limitations, conflicting priorities, and real people, things don't always work out the way the book says they will. No two situations or organizations are exactly the same. You need to tailor theories to your requirements, environment, and work style.

The SPD model shows that training program development does not have to be structured and linear to be effective. SPD is a practical, workable, and realistic approach that allows you to be flexible and creative. It allows you to work on several subsystems at a time and to move easily from one subsystem to another. SPD is empowering because you control the process rather than the process controlling you. SPD forces you to focus on the outcome—the training program—without getting bogged down in the methodology. The training program becomes the driving force rather than the process used to develop it.

SPD is not a by-the-book approach, but if you're like me—that is, interested in long-term results, but short on time, resources, and patience—this model may convince you to throw your books on linear instructional design right out the window.

Figure 2—The needs-identification form for Programmer/Analysts I

NAME _____ NO. _____

SUPERVISOR _____ DATE _____

Code	Skill description	Position skill competency	Current skill level	Training priority needed	Time frame
1100	Manages projects	A			
1200	Develops systems	A			
1300	Uses problem-solving process	C			
2100	Plans use of time	A			
2200	Applies ways to reduce stress	A			
3100	Uses Lotus	C			
3200	Uses word processing	C			
3500	Identifies PC concepts	C			
3600	Uses electronic mail	C			
3700	Uses voice-mail system	A			
4100	Does cost/benefit analysis	A			
4200	Uses performance-management process	C			
5110	Recognizes terminology	A			
5120	Recognizes basic concepts	A			

N—Not needed (the skill is not required for the job.)
A—Aware (should know a little about the skill.)
C—Confident (able to use skill on the job.)
P—Proficient (able to perform at an advanced level.)