Performance-based Training Is a Little Harder to Do

By Peter R. Hybert

Introduction

There are a lot of people that do “performance-based training.” At least, there are a lot of people that label what they do as “performance-based training.” But, if you look at actual work product (or descriptions of it), it is apparent that a wide range of criteria is being used to apply that label.

Having designed and developed performance-based training for more than 17 years, I have noticed the following problems with training that is intended (and even believed) to be performance-based.

- Mistaking learner activity for performance
- Creating performance-based application exercises that are too unlike the job performance to be effective

These two problems are serious. If the goal of training is to enable people to go back to the job and do something, failing to build training that really teaches performance is a defect—it is a product that does not do what it is intended to do.

There are good reasons that the problems above occur, but there are ways to avoid them as well. In a lot of ways, building and delivering performance-based training is more difficult than building skill- or topic-based training. But, in a lot of ways, it is actually *easier* to do performance-based training.

Like anything, the key is first realizing there is a problem (or a better approach) for getting up the learning curve, and then dealing with the effects of making the change. For example, going performance-based might identify that the current instructors, though expert in a subject, are not able to teach how to do the job. Or, you might need to develop more scenarios/simulations as exercises and find that your developers lack the expertise. But, with a plan, almost anything can be done as long as the rationale is sound. The rationale for performance-based training is sound.

What Is Performance-based Training?

Performance-based training teaches the job performance. If the performance is to troubleshoot a malfunctioning computer, the training should teach people the performance. The performance is not “knowing all about each part of the computer”—it is about identifying and clarifying symptoms, recognizing potential causes, diagnosing the problem to identify and verify specific causes, etc.

A performance-based approach would include plenty of hands-on exercises doing troubleshooting. And, the hands-on exercises should be as similar to the real job as possible. If part of the job includes talking with a user to clarify the symptoms before starting technical troubleshooting, then the hands-on should include a role-play element. And, the script/setup for the role-play should include “fuzzy” and even inaccurate information (i.e., use perceptions of the problem) so it is more like real life.

Analyzing Performance

The first step in designing performance-based training is figuring out what the performance is—analysis. It starts with "chunking." All jobs, processes, and roles can be defined by the major duties or areas of performance and, within each large chunk, details such as the outputs produced and the tasks needed to produce them can be identified in context. In addition, you need to define key performance measures for the outputs and what typically goes wrong.

For example, if it is important that the troubleshooter finds the right problem and does it quickly, some of the things that could typically go wrong might include taking too long or solving the wrong problem. Maybe sometimes the performance problems are rooted in a lack of knowledge/skills. But maybe sometimes the problems are due to poor input from the customer. All this means is that the performance is more complicated than it might seem on the surface. The training needs to reflect that and provide practice that is more like the real job. This is not hard to figure out—you can ask the master performers.

Designing the Solution

Understanding the performance is only the first step. The next is figuring out a way to teach the performance. Sometimes a training designer can make this more difficult than it needs to be by trying too hard to "add value." We can cut the performance into a million enabling knowledge/skills and spend a whole bunch of time developing presentation materials, exercises, and tests for each individual knowledge/skill. Unfortunately, many trainees have problems trying to put together a bunch of separate pieces of learning.

A better way to approach this is to start by targeting a simple case of the integrated performance. Continuing with the troubleshooting example, give them some basics and then give them a simple situation to troubleshoot. Then debrief and build on these basic skills by addressing things that help them solve more complex problems. For the learner, it is better to learn how to do the whole performance going from simple to complex than to learn all kinds of enabling detail and never have a chance to put it all together. Again, it is easier to design performance-based training than skill-based training—you simply have to define instances of the performance and incorporate simulation exercises around them. Then, intersperse simulation rounds with brief lessons on enabling skills.

Developing the Solution

Once you enter the development process, performance-based training can actually become more difficult than skill-based training. Detailed design and development of simulations can be difficult. The intent is to mimic the performance as closely as possible but without wasting resources trying to emulate things from the job that aren't necessary for performance. You have to make the simulations challenging enough to be realistic, but still carry clear teaching points.

Building a simulation requires time working with master performers. You have to build simulated job information and artifacts (such as customer accounts, reports, even simulated systems or equipment). On the other hand, most of this exists in the job environment (the primary exception being the equipment) so, if you have developed contacts in the master performer world during analysis and design, you can usually get samples from the real world and just modify them as needed to fit the training design.

Delivering the Solution

Delivering performance-based training is also both easier and more difficult than skill-based training. On the easy side of the equation, as the facilitator, you don't have to be as entertaining! The intent is for each participant to get a little input and then apply it—as facilitator, your job is to set up the simulation, "facilitate" it (i.e., make it run smoothly), and debrief it. The participants do the majority of the work!

But, this can be more difficult as well—to really build skill you have to get the participants to work. They have to execute the performance more than once (otherwise, how can they improve?). You have to be able to provide useful, individualized feedback and suggestions for improvement. You have to "cheerlead" the group to get them to play along with the simulation (including looking beyond things that are not exactly like the real job and focusing on the learning intent).[1]

Conclusions

If your goal is performance, then performance-based training is the way to go. Re-orienting the training toward performance starts with a subtle shift in perspective and ends up in a dramatic change in the training program. In many ways it is more difficult and it is also simpler. It is consistent with all the adult learning and instructional design principles but much different from how many of them are operationalized in practice. The key is to think about how people really learn—they learn by trying things and then figuring out why it didn't work as well as they hoped. Performance-based training gives the participants an opportunity to try out their learning and make adjustments until it works. From a reductionist perspective, you are just condensing rial and error. But, trial and error does eventually lead to performance. Too often, skill-based training does not.

An earlier version of this article was published in CADDI's newsletter, "Pursuing Performance," in 2001.

[1] As an aside, I once facilitated a simulation-based course for product managers in which the simulation consisted of cross-functional product team meetings to plan fictitious product development and introduction. During one delivery, a participant suggested that, because in the real world they spend a lot of time coordinating meetings (e.g., picking a date where everyone is available, reserving a meeting room, etc.), we should include a simulation on meeting planning. He even suggested that participants leave the classroom so they would have to physically call each other! Needless to say, we did include a brief lesson on coordinating meetings, but it would have been way too much work to build a simulation to teach this as compared to the relative training payback.