Chapter 1

General Rapid Development Techniques

As we've already said, most rapid development techniques depend on the type of delivery system you have determined will work best for your training. However, some are effective in a number of deliveries, or are just so darn basic to the development process that they are the good old "no brainers."

NEED TO KNOW IS WHAT'S NEEDED

One of the most basic of these, although at times even the best course developers get off track and forget it, is the concept of "need to know" versus "nice to know." There is usually a great deal of content that might be covered as you develop your material, and usually not enough time to cover it all, particularly if you plan on adding some of those pesky but so important learner exercises and activities. Even if you did a good job of creating tight objectives back during the design phase, you still need to differentiate between what a learner must really know to be successful in meeting those objectives and what is interesting, but not critical.

Keeping your material development focused on only this "need to know" information means that you don't waste time developing material that you will delete later during the pilot stage when you find you have no time to teach it and realize that it really wasn't all that terribly important anyway. You couldn't ask for a better rapid development technique than simply not developing materials because there is no need to, and this works no matter what delivery process you are using. Stick to developing what the learner really needs to learn (and not what you think is "neat"). You'll never go far wrong, and you'll go right more rapidly.

DON'T REINVENT THE WHEEL

You've heard this old adage before, but it is a really effective rapid development technique. Before you develop your own materials, search around for best practices, for stuff that has already been developed and that you can appropriate (legally. of course). In these days of the web, communities of practice, and even blogs, there is a lot of good material already developed, just sitting out there waiting for you to make it your own. Some may require a little alteration, but that takes a lot less time than developing it yourself from scratch.

As an example, for a program I recently developed for a client on cultural sensitivity, which I knew very little about, my research led me to a website created by a wonderful expert on the topic. He had objectives, an expanded outline, and even a few activities that had been proven successful. I had my clients annotate the outline I retrieved from this website, indicating those areas they thought most important, revised the objectives to reflect their thoughts, and then used the revised objectives, outline, and activities to develop course materials specifically related to the clients' needs, in about one-quarter of the time I would have spent doing it in the usual way, with task analysis and subject-matter experts, followed by laborious material development.

There is plenty of good stuff (and unfortunately plenty of bad stuff as well) available out there that can help shorten your development time, if you only take a little time to search for it. Do a Google search, do an old-fashioned literature search, ask a blog or a community of practice. Forget the "not invented here" syndrome and use this material as a whole, or just pieces; but to take advantage of it, you have to look for it.

WORKING BACKWARD

Another general rapid course development technique that can work in almost any situation is to start at the end, at least the end of the development phase, and work backward. In this case, the question you should ask yourself is, "What do I need to deliver when I implement my course?" There are a number of ways to answer this question, although the fullness of the answer will most often be based on the amount of design work, or at least design thinking, you have done. The answer might range from a brief (very brief) statement of the content, to a formal series of goals for the training, to simply a declaration of the title and the delivery system, for example; "A Web-Based Program on Leadership."

This answer, short as it is, provides you with the two most important things you need to know in the development process, what

the course generally entails and how you will deliver it. Arguably it doesn't tell you a lot about either of these key factors, but it is enough to get your development started.

The type of content (leadership, management skill building, technical skills development, familiarization, such as in an orientation program, computer skills, etc.) tells you about what your training content will be like, the objectives you created during design phase providing the details. Meanwhile, knowing the delivery system puts you on a particular development pathway, automatically giving you data such as the need for trained facilitators, depth of material development, type of visuals required, evaluation formats that you can utilize, and much more. With the answers to these questions in hand and the products of a well-done design phase, you have enough to begin course development or to find already developed materials that will work for you.

WORKING BACKWARD: PART TWO

A second rapid development technique that also involves working backward is one we've already mentioned, starting with the learner evaluation and developing the materials based on it. This is a lot more complicated than the previous shortcut—and more complex than it might seem at first glance. First, you need to have a learner evaluation completed, and it must be done properly, that is, criterion based and validated. If your design process ended with the production of the objectives, you will need to create the learner evaluation before you can proceed. You can still use this approach as a rapid development technique; it's just not quite as rapid.

The main advantage of developing materials based on the learner evaluation is that you create only "need to know" materials, and in the form required by the evaluation. Obviously (we hope), if it's on the test it's important, so that means it's "need to know."

The form aspect is a bit less clear to see, unless you have developed a significant number of learner evaluations and understand the difference between cognitive and performance evaluations. Using the normal "work forward" approach, developers often add a lot of cognitive material to a learning process that is mostly performance based. This material, while it seems important, is usually "nice to know" and less critical to the completion of, and so-called learning of, a performance.

For example, it might be nice to know how a mercury thermometer works, but if the task to be evaluated is to determine the

temperature of a non-miscible solution, knowing how to mix the solution, take the temperature at the boundary layer, and read the thermometer are much more critical.

A good performance evaluation for this task would require the learner to take the temperature properly and probably ask nothing about how a thermometer works. Developing backward would allow you to recognize this immediately, while developing forward might mean that you never catch this inclusion of unnecessary material at all, or at best not recognize it until you've spent the time creating nice time-consuming diagrams and PowerPoints of how a thermometer functions.

DEVELOPMENT TEAMS

Developing materials through a team approach can be an effective rapid development technique, if you have a team. A development team is almost a requirement in asynchronous web-based learning deliveries, which we will discuss later, but it can be used with some success in other deliveries as well.

One common team approach is to have someone who is good at graphics, or a real PowerPoint enthusiast, do the visuals, while others develop the print content, evaluations, etc. Another less common team approach is to break the program into modules and give each team member one or more modules to develop separately. It doesn't even matter whether the team is in the same location, as computers and web meetings allow them to work and meet from just about anywhere.

If you use this rapid development technique, be sure you have a strong style guide so that everything looks the same when you pull it together and a development checklist to ensure that every team member develops all the pieces that need to be developed. Having templates helps in team development as well, as does planning development milestones so that everyone stays roughly within the same time frame. It's not much of a rapid development technique if 80 percent of your program has been developed but one of the teams fell behind and will take three additional weeks to finish.

Speaking of style guides, having a good one is another rapid development technique all on its own, not only in team development, but anytime you don't want to waste hours thinking about the basics of how a slide should look or what format you want your participant package to come out in.

I've seen all sorts of style guides, small ones, huge ones, ones based on this model or that, and my best recommendation for using

one as a rapid development technique is to keep it as simple as possible. Add to your minimal style guide a series of templates for the various end-products of whatever delivery systems you use, and you have a eminently useful document, as well as a great rapid development shortcut.

VENDORS AS RAPID DEVELOPMENT SHORTCUTS

This is another no brainer, though it is one that comes in a few different flavors, depending on your development needs. Basically, if you can afford to have someone else develop your material, you don't need to spend the time doing it yourself. "Duh!"

Those flavors I mentioned though make this rapid development technique a bit more interesting than simply outsourcing. The "plain vanilla" flavor of using a vendor as a rapid development technique is to find someone who has already produced content that matches your objectives and buy it. Even if the content is only close, buy it anyway, and modify it to meet your needs. Be sure to ask the vendor for permission before you do.

A more exotic flavor is to use a vendor's knowledge of the content, how to develop materials, or both if you find the perfect one, to literally develop custom materials for you. The right vendor should be able to develop your materials faster, better, and cheaper than you can. I know the common wisdom is that you can't have all three, but if you find the right vendors they should be faster than you alone as they will usually have a team available to work on your project; better, as they may have experience with the content that you don't, and certainly more experience in development; and even cheaper, as we sometimes forget to factor in all the ancillary costs that come with material development, such as subject-matter expert time, review and pilot time, copy costs, binding costs. . . .

The right vendor can start work immediately, won't be pulled off the job to work on something else because the CEO came up with a "special" idea, and can add more resources more quickly if needed than you probably can, since the vendor doesn't need to go through all your hiring procedures. The key to making all this work, of course, is constant communication between you and the vendor, in both directions.

Now you might be thinking that using vendors/consultants to create a custom training program isn't such a novel idea, but remember that we are talking materials development here, not an entire instructional design. In this model you are still responsible for the

analysis and design work, as well as the implementation and evaluation of your program. You're only requisitioning help in developing the learning materials, whatever they may be, which is what development is all about. Because you will use your own expertise and corporate knowledge where it is most effective, during analysis and design, the process will be a heck of a lot cheaper and more efficient than hiring a consultant to create an entire program for you from start to finish.

A slightly different flavor of vendor-based development shortcut is to hire only the vendor personnel you need for specific purposes, for example, to develop a specialized simulation that they have the experience in but you do not, or simply to do your graphics, or even final editing. There are many specialized resources available to you as you develop your course, and hiring any one of them can be a time-saving rapid development technique.

A more specialized flavor is to use vendor instructors to facilitate your program, particularly if they helped create it or the materials that are in it. This might not be a true rapid development technique, as instructing is part of the implementation phase of ISD, but the training of instructors to facilitate a particular course comes under development, as we will see in the next chapter. If you can hire trained instructors from a vendor, ones who already know the content and are well-experienced facilitators, you've saved development time that would have been spent training your own instructors, and so you have another rapid development technique.

An even more specialized example of this type of rapid development technique is exemplified by a situation I found myself in once as a training director. My instructional designers and I had developed a leadership program with the help of a vendor we had worked with often. When we were getting close to piloting, the vendor's project manager came to me and opined that my plan of having my designers, who were also good facilitators, facilitate the program might be questionable. He said he didn't believe they had the background and content experience to teach this course.

After some further thought and a rousing "We agree" from my designers, I realized he was right. My people had never been managers and would have had a difficult time relating to some of the management processes we had developed for the course. I took the consultant's recommendation and hired two of his top-flight management trainers to be our facilitators. It would have taken me many hours of coaching to ready my staff for the facilitation, while

his facilitators came right in and went to it. Of course, I had my folks sit in on a few of the classes, and over time they became confident enough to teach it themselves, but that's not really a rapid development technique, and is certainly another story.

Here are some characteristics of top-shelf vendors/consultants that might help you make up your mind if you're thinking of using one as a rapid development shortcut.

Top Vendor/Consultant Characteristics

- Has a good track record with references
- Gathers plenty of information on your needs and your company before recommending a delivery process or even a piece of one
- Is proficient in a number of authoring products
- Uses the most up-to-date software
- Uses sound instructional design
- Defines things the way you do
- Has the same high standards that you do
- Uses technology that is readily available
- Has a strong development team
- Provides good customer communications
- Knows the steps that lead to the completion of your project
- Provides a quote that is not too high or low compared to others
- Does custom learning development as main function, not a sideline
- Is flexible
- Is located where it is best for you, not for them
- Will spend as much time as you require on-site.
- Has web conferencing facilities if needed
- States that rights to the program are solely yours
- Can provide maintenance and revision services if needed
- Can handle translations if required
- Doesn't tell you that he or she can turn your trainers into developers in practically no time
- Has trained instructional designers on staff
- Exhibits the creativity of the staff

- Has experienced storyboard developers if required
- Has experienced graphics designers if required
- Has an in-house programmer with excellent skills
- Can do ROI and CBA for you if requested
- Is "comfortable" for you

GAMES, A SPECIAL RAPID DEVELOPMENT TECHNIQUE WITH A SPECIAL RESOURCE

One rapid development technique that is useful anywhere, but is for the most part associated with classrooms, is games. Games can be used as activities after the content has been presented or they can themselves present significantly large blocks of content when done with a strong debriefing. If you need to develop specialized games, you tend to lose some of their gain in development time, but if you can use games that have already been developed, and into which you basically insert your content, the development time savings can be substantial. Some of these games even have built-in scorekeeping and timing mechanisms. My friend and esteemed colleague Thiagi terms these types of games *Framegames* and has any number of them for you to borrow and make your own on his website, Thiagi.com.

While best known as classroom materials, games can also be used as a rapid development technique in asynchronous web-based learning, and there are a few now appearing in synchronous web-based deliveries as well. These delivery systems and their attendant rapid development techniques will be discussed in more detail in later chapters.

If you need to develop your own games Learningware and Games2Train are two popular vendors of game development tools.

REPURPOSING

One of the more common rapid development techniques is repurposing, that is, taking a course that was created for one type of delivery and redeveloping the material for a different delivery. Repurposing became the concept of the moment early on in computer based training, and again when synchronous web-based training first appeared on the scene. Many vendors touted their ability to take your classroom programs and turn them into computer based, or later web-based programs. Some did this very well, others not so well. We will be discussing repurposing of various development pieces and parts all through this book, so we'll not detail it now, but

here is a fairly extensive list of items to consider if you are thinking of using repurposing as a rapid development technique for creating asynchronous web based training from your classroom materials.

Questions to Ask When Considering Repurposing

- Will the end-users be comfortable with the new delivery format?
- Will their supervisors support the end-users during implementation?
- Will the current instructors support the course?
- Does the current course require a change of delivery?
- Is standardization or consistency important?
- Are there high-level management sponsors for the conversion?
- Is the current delivery taking the learners away from the job for too long?
- Do the learners say the course is too long?
- Do you need to train large numbers in a short time?
- Is the continuing target audience large?
- Is there convenient hardware access?
- Are qualified instructors hard to find or keep?
- Will this repurposing reduce transportation and housing costs for learners?
- Is the content strongly cognitive?
- Is interpersonal interaction required to learn the content?
- Is hands-on equipment required?
- Can simulations be used effectively?
- Is frequent practice required?
- Is behavior modeling required?
- Are the course objectives not being met with the present delivery?
- Does the required software reside where it is needed?
- Is there a budget for things that do not currently exist?
- Do learners in the course vary widely in ability?
- Does the course target entry-level employees?
- Are the learners self-directed?
- Is documentation and record-keeping a problem?

- Is recertification required?
- Is tracking of learner progress important?
- Is there need for course security?

OTHER GENERAL RAPID DEVELOPMENT TECHNIQUES

There are any number of other general rapid development techniques, some of which are more efficient for one delivery or another, but all of which can be effective in various delivery environments. Here's a quick hints list for how to use some of the most effective ones:

- *Use prerequisites to decrease the amount of content you have to create.* This is actually more a design process, but by deciding that the learners will be required to come to the training with some basic knowledge (and letting them know that in advance, don't forget), you will be able to reduce the amount of content development.
- *Use your expanded outline as an instructor guide, combined with the very minimum of learner materials, and keep revising.* This hint is from the "you'll never get it perfect anyway" school of thought. Start with the least development that you can and continuously upgrade the materials every time you do a delivery, based on what you learned during that delivery. You can even add ideas generated by the learners during the class to your materials.
- *Utilize materials that were not created specifically for instruction*. This might include books, chapters from books, magazine articles, websites, manuals, videos, corporate websites, even television programs. I once used excerpts from the various "Star Trek" shows to discuss leadership skills and saw a wonderful example of a person using positive approaches to giving negative feedback on a show called "Project Runway" that I can't wait to try out in a management class. We'll discuss some specific uses of non-instructional materials as rapid development techniques in later chapters, but such pieces can save you development time in almost any learning environment. One caveat to this technique though: Be very aware of the copy-right laws.
- *Use computers*. This may seem like a blinding flash of the obvious in this day and age when computers are so pervasive, but we often miss some of their possibilities. Check out the specialized

development software available to see whether it can reduce your development time, think about how general software such as that used for flowcharting or creating spreadsheets might be effective, or create your own computer-based templates for your material development. Even office software like PowerPoint and Word can reduce development time when used effectively.

- *Use subject-matter expert focus groups for reviews.* Instead of sending your materials around to various subject-matter experts for review, hold a focus group review where all the subject-matter experts meet together This not only gives you better control of the time they take to do reviews, but also allows them to all agree on a final version of the materials without the materials going back and forth among the SMEs a number of time-consuming times.
- *Borrow from the best*. When you see a program that you like and that you think worked really well, do a quick outline of how it worked (not the content but the instruction), then develop a file of these outlined ideas that you can reference when developing your own material. You can do this for learner materials you come across that you feel are "top shelf" as well, outlining how the materials were made or simply noting what it was that you liked about them and how you might use them.

RAPID PROTOTYPING

There is one more rapid development technique that we should mention, a concept known as rapid prototyping. Rapid prototyping is one of those slippery concepts that sounds like a good idea when someone talks about it generally, but when it comes down to actually explaining how to do it, you find it hard to express, as it has a number of permutations. That's not to say it isn't a good idea; it is, but it's just sort of hard to describe.

The theory behind rapid prototyping is to recognize that your end-product is simply too complicated to try to make it perfect the first time, before anyone sees it, or the second time, or even the third. Instead, do a quick example of what you are planning to accomplish and see how the decision-makers and others react to it. Then make corrections and go on from there. Now this goes against the grain of almost every instructional designer ever born. We all want our programs to be perfect before anyone sees them. However, in rapid prototyping the 80/20 rule is in effect, and close is more than good enough.

This approach was first used in a training environment by developers working on computer-based learning projects where finding out that the client wasn't pleased after you'd completed all of the programming and graphics was an invitation to disaster, not to mention a waste of large amounts of development time. Instead, they would create small pieces of the final project that exhibited aspects such as screen design, learner interface, activities, content, etc. Then, as the client saw and approved these, they could go on and complete the program, incorporating those characteristics that passed the test.

A slightly different concept of rapid prototyping is to take a small piece of the whole program content and finish it completely, including the interfaces, visuals, testing, basically creating a small working model. After approval of this model the other pieces are created and assembled.

Still a third notion uses the above approach, but for each program piece. As a piece is completed, it is considered and either selected or rejected. Selected pieces are assembled into a larger piece, and that piece goes through the same selection process until the final program is complete. Rejected pieces are kept in a file for possible use in another program.

Some rapid prototypes are created simply to obtain agreement on color schemes, fonts, and screen characteristics, others to exhibit interfaces and navigation. When expressly created for theme or color decisions, they are often done using static graphics software instead of the delivery software, which is more difficult to program, thus saving even more development time.

There are possibilities for rapid prototyping as a rapid development technique in most of the major delivery systems, although it obviously is most effective in asynchronous web-based programs. As with all our general rapid development techniques, keep it in mind when you are planning any major program development, and use it whenever it will work for you.