



## Learning-By-Doing, Good and Cheap

### *Building Quality Simulations, Affordably*

More and more learning designers are recognizing the benefits of performance-based e-learning: it's engaging, learning gains are high, users like it, and it can treat subjects difficult to teach via conventional methods. However, simulations generally take a lot of time and money to build. Why is that?

#### **1. They contain a great deal of content.**

The longer the simulation, and the more freedom allotted the user, the more content that has to be developed. This tends to go up exponentially: for a simulation consisting of a sequence of "turns," supporting 10 turns with 4 possible user choices per turn means potentially well over a million system responses would have to be crafted! And, the more content there is, the more the potential for mistakes, so testing costs go up as well.

**2. They're tricky to develop.** Good simulations offer meaningful choices to users, informative feedback, and relevant coaching every step along the way. It takes specialized skill to author all of this, skill that you don't find "off the street." Moreover, in a simulation that offers users freedom to complete tasks as they wish, authors are not composing a simple, linear narrative, but are instead composing a tapestry of narratives—akin to writing several novels simultaneously.

**3. They require collaboration by expensive resources.** The best kinds of simulations combine an exploratory environment (where users can perform as they do in real life) with just-in-time

coaching. However, this takes a deep understanding of the subject matter, of cognitive science and learning design, and of the user population. Is the simulation "real?" Is the coaching accurate? Are people learning the right things (and what exactly *are* they learning)? It's rare to find a single person who can answer all these questions. Hence, a team of people need to be involved, and often inefficiently – SME's spend time copy-editing, or simulation authors help refine the behavioral model. So, costs mushroom.

#### **Suboptimal Ways to Reduce Costs**

To overcome these obstacles, e-learning vendors often constrain their designs in order to contain costs – they limit the amount of content in the simulation, or simplify coaching, or reduce personnel. These constraints, while not inherently flawed, often compromise quality.

For example, many build "linear" simulations. In each turn there is one correct choice that moves you to the next turn, and the other choices bring up coaching that gives feedback and sends you back to try again. This cuts down on the amount of content that needs to be developed, but often at the expense of engagement – users aren't free to proceed as they wish, so the experience doesn't feel authentic. And, they may limit learning potential – if users can't make the mistakes they make in life, chances are the coaching will have limited impact on their behavior.

Another means to reduce costs is to reduce the amount of coaching that gets

provided, or to use “static” coaching, that is, to coaching that is not written to address specific users actions in the simulation, but rather, treats subjects in a general way. This is often seen in physical simulations of processes, such as training on equipment, where “coaching” often consists of on-line operating manuals.

The problem with this approach is that it robs the simulation of learning potential. When users make mistakes, or ask for feedback, coaching is either lacking or too vague, irrelevant, or off-task to be of much use. And if the simulation doesn’t teach, what good is it, even if it was built cheaply?

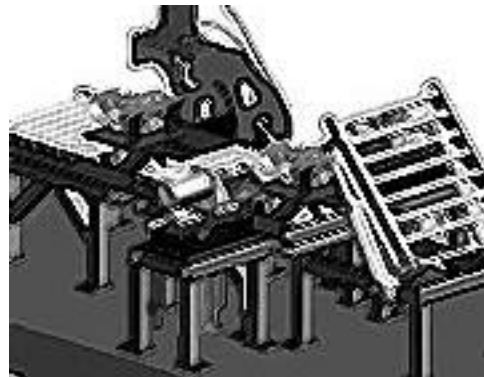
A third method to control costs is to outsource production to external writers, using experts to set the initial objectives and to review the final piece. However, it’s difficult to find writers with the talent to build a simulation in isolation, without continuous feedback from people “in the know.” Left alone, writers often commit the single biggest mistake that a simulation developer can make, which is to simulate for the sake of simulation – that is, to include aspects of reality simply because they’re part of the world being modeled, not because they contribute to learning. Good simulations come out of continuous dialog between SME’s, who understand the behaviors being taught and how they are learned, and developers who know how to capture reality in a simulation. If one or the other takes over development, the simulation suffers.

Another way vendors hold development costs down is to try to spread costs over many simulations. With this approach, the resources and budget are allocated to produce a good simulation which is sold off-the-shelf as many times as possible in order to recoup the investment. The problem with the one-size-fits-all approach is the loss of reality for many learners and, potentially,

dilution of learning: having learners role-play selling copiers, for example, may not translate very well when they’re trying to sell e-commerce solutions on the job.

In contrast to these methods, which limit costs by reducing quality, there are other techniques you can employ to lower production costs and raise the quality of simulations, without sacrificing the components that go into them.

### *1. Develop Via an “Assembly Line”*



If you simply send an author off to build a simulation, he/she will almost certainly fail—there’s simply too much to think about, all at once. However, if you instead break production into a number of steps, and build a development/review process around each step, there’s a much higher probability that the writer will complete the task, and create a superior product as well. This is the idea of an assembly line, and is a powerful technique for developing simulations.

For example, here is a production plan for simulation development that involves a number of small steps:

<b>Step</b>	<b>Outcome/Rationale</b>
1. Identify the learning objectives for the system	This step ensures that all members of the design team share a common vision for what the simulation is supposed to accomplish. The decisions made in this step will influence all subsequent decisions about

- 
2. Develop a list of specific behaviors to be taught  
what to simulate and how. Simulations are about learning-by-doing, so the team needs to focus on behaviors. This step identifies the ones to be addressed by the system.
  3. Identify tasks to simulate that involve the behaviors, and assign the behaviors to each.  
This helps ensure that all of the desired behaviors are being addressed, and also helps decide what needs to be simulated for each task.
  4. Develop a "correct path" for each task (a sequence of steps that an expert practitioner would take to complete the task).  
Here, writers determine what will need to be simulated, and are building a skeleton that other simulation content can ride on. The correct path is fairly easy to write and review because it's linear. It ensures everyone agrees on best practices and what range of actions a user will be presented in the simulation.
  5. Develop alternate actions that will be made available to users at each step in a task, and develop alternate paths representing alternative strategies a user will be allowed to pursue.  
Here is where writers address the various misconceptions and mistakes made by the target population, which lie at the heart of a simulation's effectiveness. Because the correct path and behaviors have already been agreed on, this task is vastly simplified – writers can use the correct path as a skeleton off which to hang alternate actions.
  6. Devise coaching and assign to each user action in the simulation.  
Once the content of the simulation is written, it becomes fairly routine to write up coaching and assign it. SMEs can be involved to help write the coaching to the review how it's assigned.
  7. Produce media (audio, video, etc.) can be developed and incorporated into the system, and alpha and beta versions rolled out for review and

8. Copy edit, perform user, alpha, and beta testing  
This phase should go fairly quickly since most of the significant decisions were made earlier in the process.

Similar to the assembly line approach is rapid prototyping and phased rollout -- that is, to develop one user task, and then set it up for Q/C and SME review while the next task is worked on. This enables various members of the project team to view the system early in the process, and it will surface any errors or omissions in time to avoid replicating them in subsequent work.

The main advantage to a production-line approach is that it turns a difficult task into a series of easier ones. Even more importantly, it facilitates participation by various members of the team in some of the day-to-day decision-making.

## 2. Modularize the Content



A second approach is to divide the content of the simulation into discrete "objects," which can be mixed and matched to produce the final piece. For example, you can develop coaching objects, media objects, user action objects, and system response objects, and so on. The idea behind such "objects" is that they can be reused over and over, which increases the richness of the simulation without increasing cost.

A clear example of this is the coaching provided to users when they make a mistake or ask for help. If each piece of coaching is defined as an object, it can be freely assigned to multiple user actions which reduces the amount of coaching that has to be written. The same holds true for user actions and system responses: if authors have the ability to assign them in multiple places, then less needs to be developed from scratch. Because most simulations involve a fair amount of repetition (a software simulation, for example, might include the same command at several places in the tasks), reusing actions and responses can make a large savings in content production.

The notion of modularization can be extended to the simulation as a whole, as well: if different sections or tasks can be modularized, they can be mixed-and-matched to produce the final user experience. For example, many vendors build soft skill simulations as a “tree:” each action leads to a unique response, and follow-up action, and follow-up response, and on down the tree. Under this scheme, actions and responses build exponentially, which means the simulation becomes large and unwieldy, very quickly. An alternative is to build the simulation as a “graph,” by enabling authors to reconnect sets of user actions together – for example, to allow users to deviate from the “correct path” and then reconnect them sometime later. What this does, in effect, is allow authors to give users more freedom to operate, while constraining the amount of content that needs to be developed. In similar fashion, authors may build separate tasks separately, and then connect them together. The more that can be reused, the less the production cost.

### 3. *Webify production*



One of the most exciting aspects of the world-wide web, from a simulation development standpoint, is that it allows different people to be involved in production, from anywhere in the world, at the same time. If you can move production onto the web, you allow writers and SMEs to work arm-in-arm on the finished product.

For example, by building a simulation via a web-based authoring tool, different people can log in to write, edit, and review the content in the system. An author might write one piece of a task, then send an email to an SME to go through the content and add comments, or to edit. Or, several authors can log in independently to work on different components, such as different tasks. Such web-based tools can also track changes each makes to the system and the comments they enter, so that other members can review them in subsequent sessions.

This work can be augmented by web-based project management tools, which enable project managers to establish and adjust production timelines, and track progress against it.

For web-based training, the web also enables a production team to adopt a rapid-prototyping approach: the finished product is periodically brought on-line for review throughout production. For example, an author might finish a simulation task, then

publish a prototype so users can try it out. This makes it more of a WYSIWYG approach that enhances buy-in and eliminates late-process surprises.

#### 4. Automate Quality Assurance



One of the most time-consuming aspects of simulation development is performing final quality assurance. In fact, as a simulation grows more complicated, the cost of quality assurance can eclipse that of the product itself, as more potential bugs and content flaws are introduced into the system and it becomes more difficult to systematically check for them.

One solution is to automate some of the quality assurance process. For example, automated tools that look for broken links, malfunctioning interface elements, and so on, can save many man-hours of testing. Spell- and grammar-checkers can weed out typos. Conceptual tools can spot inconsistencies in the simulation – for example, “loops” in the logic of user action chains in a simulated task (such that a user might be able to cover the same ground repeatedly).

However, there is no substitute for a human being to look over the content, and this aspect of production can benefit by providing checkers with a means to look over the simulation in a systematic way. Even something simple like the inclusion of a “back” button can help in this regard. On-line commenting, which allows testers to

make annotations to specific user actions at specific times in the simulation, can save time and money, and catch problems while they are fresh in the tester’s mind.

#### Summary

These four techniques can together bring down the cost of producing rich, engaging simulations to the point that they’re affordable by everyone. And, they can do away with the need for a one-size-fits-all approach, allowing simulations to be customized so they fit each application and ring more true to the learners. The idea behind these techniques is to standardize components, and break larger ones down into smaller ones, so as to make production as simple, streamlined, and conceptually coherent as possible. By lowering costs and speeding up production cycles, these techniques should help simulations become more integral elements of e-learning curricula, as they rightfully should be.