



## Customizing Simulation-Based Training

When buyers view a training simulation, they usually ask whether it can be customized—to include their products, or their processes, or their terminology, or their best practices. After all, the whole point of simulations is to allow learners to apply their new knowledge in actual job situations. If those situations don't reflect the buyer's reality, how effective is the learning?

Given the complexity of most simulations, however, customizing them to any significant degree is cost prohibitive. One way to overcome this is to convince the buyer that a simulation doesn't have to mimic their business to work. Another is to offer to make minor alterations, like adding the buyer's logo or information about their products.

Alternatively, you can address the customization question head-on, by developing a means to do it quickly and profitably. It takes advance preparation, but can be done.

### Why it Costs So Much

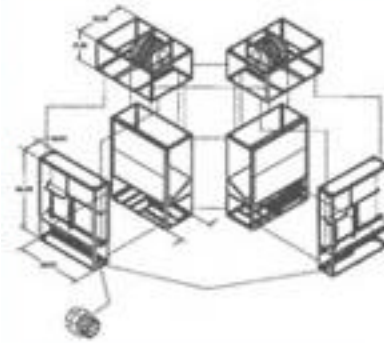
The first step in containing the cost of customization is to determine where they come from.

Obviously, the more that needs changing in a piece of software, the higher the cost. With conventional e-learning, you can edit a page here and there and leave the rest untouched, so cost rises linearly with amount of customization—re-writing 10 pages of material costs twice as much labor as re-writing 5 pages, for example.

Most simulations, in contrast, involve so many interdependent elements that production cost rises *exponentially* with the amount of customization. Let's say you're customizing a sales simulation involving face-to-face interactions with a simulated character. If you edit a couple of the customer's responses to reference a company's product line, you then have to edit the user actions that precede and follow it. This may require editing other actions in order to maintain the flow, which means editing still other responses. You probably then have to adjust the coaching that goes along with the changed actions. If you're using multimedia, you need to re-record audio or video. Simulations that are built a page at a time may require fine-tuning individual Flash animations or HTML pages, which tacks in

programming and quality assurance costs. As a result, editing the equivalent of 10 pages' worth of content in a simulation may take 4 times the labor as editing 5 pages, editing 20 pages' worth takes 10 times more labor, and so on. Costs quickly mushroom.

Below are several techniques for reducing the cost and time it takes to customize simulations, particularly role-play simulations.



### Modularize the Design

The primary way to lower customization costs is to modularize the various elements, so changes to one do not require changes to the other. In the above example, if user actions and character responses are kept separate from the interface, changes to them won't require additional programming or debugging time, which lessens costs. If coaching is maintained separately, it may not have to be changed whenever a user action or response is changed, which again reduces labor. If actions and responses are written in a way to minimize dependencies on what came before and what is to follow (which is a matter of clever authoring), then someone can edit them without having to change the entire narrative. If multimedia elements are isolated, different forms (text, audio, video) can be swapped in and out without having to redo the entire design.

The hard part in modularizing simulations is figuring out what the various components are and how they fit together. This takes some trial-and-error, but over time, ever-higher degrees of modularization are possible.

	$\overline{A}$		
	0	1	
$\overline{B}$	0	$AB$	$A\overline{B}$
$B$	1	$\overline{A}B$	$AB$
	$\overline{A}$		

### Make the content data-driven

Related to modularization is making the simulation “data-driven,” that is, separating the code that drives the interface from the content (user actions, system responses, etc.) that unfolds within it. This is usually achieved by maintaining content in a database or text file, which is read by the application at run-time, or “published” into HTML or Flash for stand-alone use. By separating the content from the code, the content can be changed without incurring programming and debugging costs.

The challenge in making a simulation data-driven is deciding how to divvy up the content into discrete data points; once it is done, however, customization costs drop considerably.



### Genericize the coaching

In many training simulations, coaching is offered when a user takes a wrong choice, when they ask for help, and at the end of the simulation, in a “debriefing.” If coaching is situation-specific (for example, if it makes references to particular products or business processes), then it will probably need to be edited whenever user actions are changed.

A good way to lessen authoring costs is to write coaching so as to minimize situation-specific references. The tradeoff here is that doing so may lessen its impact. One way around this is to carry two levels of coaching: one that addresses performance in a more generic way, and another that addresses aspects of a particular user action. Then, authors can edit the latter without having to edit former, which

reduces the amount of editing work. You have to devise the generic coaching to be abstract enough to re-use, while specific enough to be useful, but once you’ve achieved this customization becomes much easier.



### Create a “library” of simulations

The closer a simulated situation is to a particular customer’s business, the less that has to be customized. So, one approach to customization is to maintain a large library of potential situations, representing different industries, problems, and so on. When a customer wants a customized solution, you select situations closest to the customer’s business, then modify them for an exact fit. This is facilitated by devising a diverse array of situations, and writing each to be easy to modify further.

This approach works well, as long as a vendor has the time and resources to author a large set of situations. One of the key decisions regarding this approach is deciding what differentiates different scenarios— if you still have to make a large number of changes no matter which one you select from the library, you have gained very little. Commonly, situations are classified by industry, since companies within an industry usually share common terms, products, and processes, the things most often modified. However, deciding what constitutes an “industry” is hard. Even if you restrict it to, say, banks, community banks face different business situations than multi-national banks. For this reason, it is best to classify situations not by industry but by the kind of encounters being simulated.



### Create reusable objects

Modern-day programming is dominated by the “object-oriented” approach, which construes a software program as a collection of “objects,” each with their own data and code. The driving force behind object-oriented programming is that it enables a programmer to re-use objects (such as buttons, database readers/writers, etc.) without having to re-write the code.

The same logic holds for simulations: the more that content can be reused, the less editing you have to do. One example touched upon earlier is coaching. If coaching is written separately for each action, then an author may need to make the same edit multiple times. If, on the other hand, it is an “object” (say, a data record that is linked to multiple actions), it need only be edited once for the effect to show up everywhere.

Many other elements commonly modified as part of a customization process can also be objectified: character responses, multimedia elements, links to external sources, interface elements (such as logos), and so on.

As with object-oriented programming, this technique depends on a designer’s ability to recognize what pieces can be reused, and what can’t, which means understanding how the various pieces relate to each other.

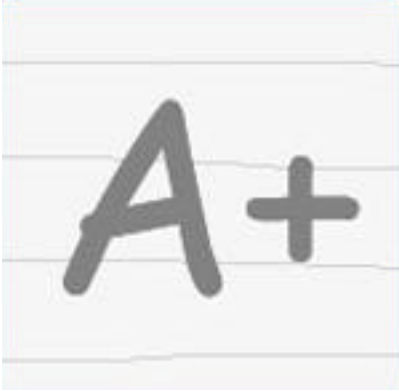


### Streamline quality assurance

One of the costs that must be factored into customizing a simulation is the labor that goes into making sure the end product performs to spec. Every tweak to the system may introduce a typo, bug, conceptual inconsistency, and so on.

Clearly, if techniques such as modularizing the design and genericizing coaching are employed, then less of the system has to be changed to customize it for a client, and hence, less quality assurance completed to ensure the system still operates as desired. Quality assurance costs can be lowered even further by creating a systematic and automated quality assurance process. For example, an automated test can ensure that all buttons and operations work as designed, without having to go through the system by hand. Spell checkers can unearth typos. Link checkers can trace through a web-based simulation and verify that all links are active. Automated asset audits can make sure your audio, video, and image files are where they need to be. Furthermore, by investigating what can go wrong when things are changed, you can focus manual quality assurance methods on the parts most likely to break after customization is complete.

Incorporating such processes may entail investment, but the return in reduced effort per customization project more than pays for it in the long term.



### Streamline the review process

If a customer is paying to have a simulation customized, they will likely want to be involved in development, to ensure their needs are met. This can mean that many people have a say in the product, including line managers, in-house trainers, senior management, users, lawyers, and more. Such involvement can drive up costs, often in unanticipated ways: multiple re-writes, interface re-design, multiple prototypes, additional support, and so on. It is challenging to figure out how to involve the client while keeping production on a fast track.

One method for channeling client involvement is to devise a series of deliverables, representing sign-off points in the process. Deliverables might include a “scoping” document, a model review document, a content mockup, a prototype, and an alpha version of the system. Each deliverable is associated with a set of decisions that need to be made. For example, a scoping document identifies what additional learning objectives need to be included in the customized simulation, and a mockup can help lock the interface design. Teasing decisions into a sequence helps minimize the amount of re-work that needs to be done.

The main thing to consider here is that most customers have difficulty envisioning a simulation in the abstract. They may only be able to provide significant input when they can see the finished product, or something close to it.

Hence, deliverables should be as close to “what you see is what you get” as possible. Otherwise, there is a much higher risk that someone didn’t understand what was being described and differing expectations among the constituents only come to light toward the end of production, when budgets and timelines are tight.

In a simulation involving multiple situations, one way to accomplish this is to customize a single

situation and have the client sign off on that, before proceeding to additional situations. Or, you might change the user actions and responses and get client feedback, and only later change the coaching to go along with it.

Then you have to be concerned with version control and logistics. How do you ensure that you are getting feedback from everyone on the same version? How do you get the deliverables out to your constituents and get their feedback in a timely manner? Scheduling conference calls and shipping documents back and forth can eat up significant chunks of the timeline, and there is always the risk that the latest version is lost or misplaced.

One solution to this problem is to “webify” the customization process. If the project constituents can view live content online, and if they can post their feedback online, there is no risk that they will be looking at obsolete material. The review cycle is significantly shortened by taking the back-and-forth volleying out of the process and eliminating much of the logistical tangle. They can review material when it suits them.

### The bottom line

The techniques above are each an attempt to isolate the changes that need to be made to a simulation to meet a particular client’s needs, so they can be made without having to change other parts. In the end, their successful implementation requires that a designer have a firm understanding of why customers would want to change the simulation in the first place, which dictates what parts must be changed, and how. By starting there, implementing some or all of these techniques becomes easier, and the net result is that a designer can deliver a better product for the customer and make more money at the same time.