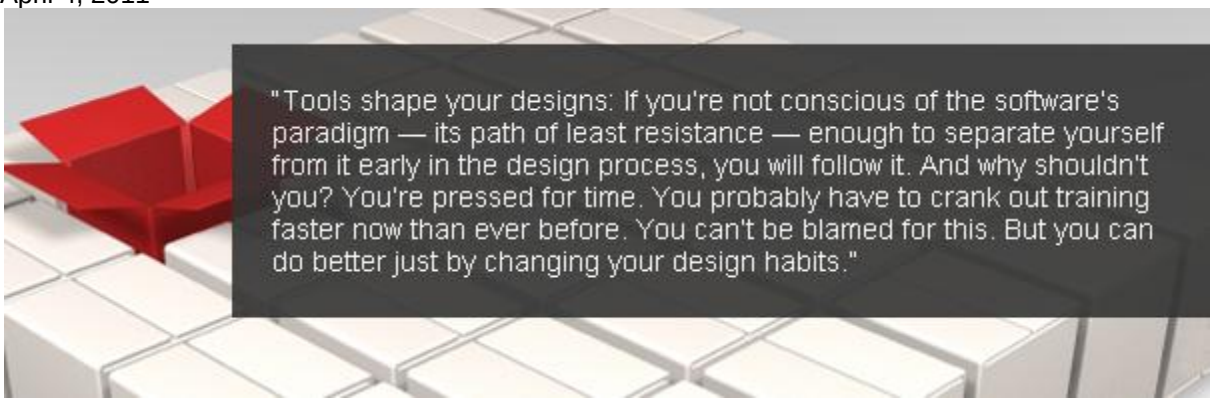
Thinking Outside the Toolbox (Apr 11)

By Judy Unrein
April 4, 2011



How much does your authoring software determine the quality of the e-Learning you create? There are a lot of e-Learning designers who say that the software you use doesn't matter. It's the design that counts, they sniff. The rest is just details.

On the other end of the spectrum, there are those who say that effective learning experiences can't be created with rapid development software. Rapid tools can't allow for enough meaningful learner interaction according to that camp; great learning experiences require heavy-lifting tools.

There must be a better way

To me, the truth lies somewhere between these extremes, and is more nuanced than either of them. I've seen great courses created with every tool that designers use, even tools that weren't designed to create e-Learning (I'm looking at you, PowerPoint). Check out Cathy Moore's famous collection of e-Learning examples. You'll see plenty of courses there that, regardless of the software their designers used, could have been created in Captivate, Articulate, or Lectora... or, for that matter, PowerPoint or Dreamweaver. But are those great, engaging courses typical of the e-Learning that makes it to today's workplace learner? Sadly, I think we all know that the answer is No. In fact, it's not even close.

And there's the rub: If rapid tools (the ones that, according to the e-Learning Guild's research, year after year, almost all of us use) *can* create engaging e-Learning why don't we see more courses like the examples above? Why do rapid development tools -- and those who use them -- still get a bad rap?

To put it simply, the buck stops with the designer. We already know the huge list of skills that e-Learning designers need to have, particularly if they are one-person shops. Well, I'm going to add one more requirement to the list: You need to have the ability to think outside of your toolbox.

To each its own (paradigm)

Here's what that means. Rapid development tools are rapid for a reason: They make it easy to create a particular kind of course, and each tool has its own paradigm.

Articulate Presenter is built on a PowerPoint backbone; Engage and Quizmaker add prepackaged interaction and quiz types. Stick with those structures, and you can have a basic, functional course in just a few hours.

Captivate is also slide-based, and also fairly easy to use for a pretty good-looking course. Starting on slide one and building a course that goes straight through to the end slide is by far the most obvious way to use the program, although recent versions have greatly expanded the software's ability to break out of the linear structure.

And then there's Lectora, which has a book structure by default, complete with chapters, sections, and pages. It's like an engraved invitation to create a page-turner... not a good thing when we're talking about e-Learning. Interestingly though, if you veer even slightly off the Template Wizard path, you'll find that Lectora's hierarchal structure, variables, and actions give it a great deal of power -- more than any of the tools I've just mentioned.

In all of these cases, you *can* create courses that are wildly off the beaten path, that are far from what the software designers originally intended. We've all used software for purposes other than its intended function from time to time, because it was the best option immediately available. For example, maybe you've used Word to edit a picture, or PowerPoint to lay out a brochure. I have a friend who is a stage manager and she uses Excel to create blocking sheets. And there's nothing necessarily wrong with that. But bending software to your purpose has costs. In e-Learning authoring, the farther you get away from what the software was designed for, the less help you get from the program — the less, in a sense, you are actually doing rapid development.

Template troubles

On the other hand, almost every piece of software has templates and wizards... the paths of least resistance. If you use them, they make it really easy to do your work quickly. Congratulations. You only spent a few hours making a course and you didn't even have to think that much. Too bad the resulting course will be forgotten the moment the LMS registers it complete. Or maybe the first course will wow the crowd because it looks better than what came before it, but what happens when the next course looks and works exactly the same? And the next? And the next? Using the "help" that programs give you is easy, but using a cookie-cutter approach is not design.

Probably the most extreme examples of this approach are PowerPoint's training presentation templates. Remember those? Here's a link to one, in case you don't know what they looked like: http://www.scribd.com/doc/49621097/Training-Presentation

PowerPoint is presentation software, so it can't be criticized too much for having a linear paradigm and limited interaction functionality. But we should be aware of the PowerPoint mindset because e-Learning designers are constantly asked to "convert" instructor-led training — mostly structured in PowerPoint — to e-Learning, giving rise to tool after tool that facilitates starting in PowerPoint and ending in Flash.

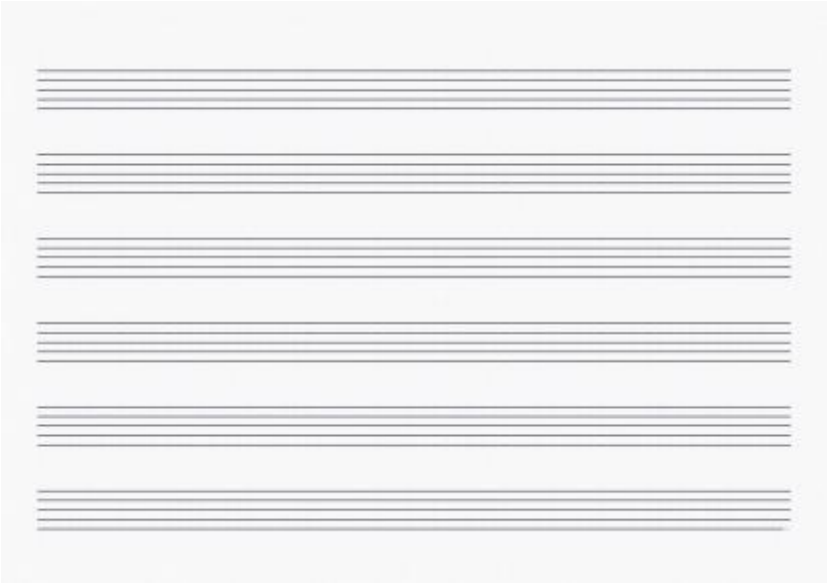
Getting outside of the box

At this point, it probably sounds like I'm completely against rapid development tools. That's far from true. My company uses Lectora, Captivate, and Articulate for e-Learning development, but you would never know that in many of our courses. Rapid development tools aren't great because they allow *SMEs* to create e-Learning (which is how a lot of them are marketed); they're great because they allow *learning designers* -- experts in learning but not programming -- to create e-Learning. The trick is to use them for development, but not let them control your design thinking.

And how is that done? Most of the tools that e-Learning designers use guide us to create page-turners. We create storyboards in PowerPoint (there it is again, influencing our e-Learning) and in tables in Word. When you start with a linear design tool, you're most likely going to end up with a linear design.

I was musing about this issue to my husband a few months ago, appreciating him as a sounding board but not thinking he would have a similar problem in his work. (He's a composer.) But he ended up showing me something surprising.

You can write music in software or with a pencil; either way, the paradigm is that of a piece of score paper:



Pretty linear, no? That may not be a problem for music -- music does, after all, start at one point in time and end at a later point, and there isn't the expectation that any input from the listener will affect the direction of the piece. But just because the final piece of music is linear doesn't mean its design process — its composition — has to be. He told me that when he starts to write, he begins in the middle of a large piece of score paper, adding on to the beginning of the piece as well as the end, adding higher parts as well as lower. He could add these things if he started at the top left, but then he would have more rework. The path of least resistance would nudge him to write from top left to bottom right, regardless of what served the piece best. This initial sketching technique allows him to write without limiting his possibilities.

And that's really the crux of how tools shape your designs: If you're not conscious of the software's paradigm — its path of least resistance — enough to separate yourself from it early in the design process, you will follow it. And why shouldn't you? You're pressed for time. You probably have to crank out training faster now than ever before, serving more people, with fewer resources. You can't be blamed for this. But you *can* do better just by changing your design habits.

A few suggestions

There are many ways to think outside the toolbox, and I hope you will post your own in the comments at the end of this article so that our industry can benefit. To start us off, here are mine:

- Get out of the software. In *resonate*, Nancy Duarte takes the reader on a process of designing a compelling presentation from the ground up. At the "Create Meaningful Content" stage, she writes, "Resist the temptation during this initial phase to sit down with presentation software; it's not time for that yet." Staying out of your development software until your design has started to form is crucial, especially if your software has a strongly-defined path of least resistance... or if you're not very experienced in resisting it.

- Sketch. Buy some big paper. It doesn't have to be an expensive sketchpad; it can be blank newsprint. Write your ideas on it and draw arrows all over, or stick sticky notes to it. I sometimes need the tactile sensation of writing and drawing on paper, but if you prefer to use software, use mindmapping or visualization software, not presentation or word processing software. Freemind is one that I like; it's open-source and available for Mac or Windows.
- Start small. Challenge yourself to create something that would be interesting even if all you had to work with was a printing press. Books are low-tech, yet many of them manage to keep us up all night, on the edge of our seats, unable to put them down; when we refer to books as page-turners, it's a good thing. E-Learning can incorporate the same strengths: a thrilling plot, compelling characters, humor, powerful writing. Books can be interactive (think *Choose Your Own Adventure*). They can even be *instructionally* interactive (see Robert Mager's *Preparing Instructional Objectives*). Rapid e-Learning can be the same.
- Start big. The publisher for a magazine where I once worked used to tell me that it's easier to start with a design that is too flamboyant and rein it in than it is to start with a design that's too quiet and try to bring it out; the latter ends up looking forced, incomplete, or awkward. I now believe the same about instructional design; it's easier to create something good when I start with a super-interactive game scenario and then pare it down to what I can actually do in my given software, than it is when I start with a bulleted presentation and "spice it up" with a little interactivity.
- Think integration. When you pare down — if you have to — think about how your tools can work together to create the experience you want, instead of limiting yourself to one tool. SWFs from Captivate and Articulate can become objects in a Lectora title — even exchanging information and commands with the title. HTML published in Lectora can become Web objects within Articulate Engage, and Presenter and Engage pieces can be placed within Engage, as well. Lectora Inspire makes it easy to integrate output from Camtasia, Flypaper, and Snagit — and includes licenses of each. You could even include a single Flash interaction in a course if there's no other way to give your learners the needed practice; that doesn't require creating the whole course in Flash.
- Don't forget visual design. Lots of rapid e-Learning falls flat because instructional designers aren't necessarily graphic designers, but a strong visual design goes a long way toward good learner engagement. I know, you don't have the budget. Look on the Internet to build your skills and your asset collections, for cheap or free… just make sure you keep it legal. Make the case for a graphic design budget on your next project by showing your stakeholders how much of an impact it can make.

Your turn

So do the tools matter? Of course they do. Every piece of software has its limitations that you cannot escape no matter how many workarounds you try. But your software's ability to create drag-and-drops or publish to Flash or store variables only goes so far toward determining your ability to create great instructional content, and even great learning experiences. The far greater piece of the puzzle – the design – is up to you.