Tools for Design and Development of Online Instruction

Bryan L. Chapman
Brandon Hall Research/Chapman Alliance, Sunnyvale, California

Contents

Introduction .....................................................................................................................................................................672
Standard Authoring Tools ...............................................................................................................................................673
Front-End Design and Automated Instructional Design Tools ..................................................................................676
Learning-Activity, Focused Design Tools (Simulations, Games) ..................................................................................678
  Software Simulation Tools ....................................................................................................................................679
  Soft-Skills Simulation Tools ..................................................................................................................................679
  Hard-Skills/Technical Simulation Tools ................................................................................................................679
  Gaming Tools .........................................................................................................................................................680
Team Development/Publishing Tools .............................................................................................................................680
  Interactivity Templates ..........................................................................................................................................681
  Dynamic Pretesting ................................................................................................................................................682
  Learning Objects ....................................................................................................................................................682
  Storyboard Specifications ......................................................................................................................................682
  Links between Assessment and Competencies .................................................................................................682
Conclusions .....................................................................................................................................................................683
References .......................................................................................................................................................................684

Abstract

Today’s instructional designers can choose from a wide variety of authoring and online learning development tools. With the evolution of learning technology, expectations from outside the learning and training discipline are looking to the technology to provide instructional design guidance and allow novice developers to create instructionally sound online learning. This chapter explores the tools that instructional designers and novice developers use most often to create online learning from both a design and development perspective, the types of instructional design support that can be found in different types of development tools, and how these tools may (or may not) impact the ability to meet instructional needs. A classification system is used to
analyze tools as (1) standard authoring tools, (2) front-end design and automated instructional design (AID) tools, (3) simulation and gaming tools, or (4) team development or publishing tools such as learning content management systems (LCMSs) and groupware authoring. Although instructional design guidance, in some form or another, is now embedded inside several classifications of development tools, considerable innovation and research are still required to create holistic systems that negate or minimize the need for instructional design expertise, allowing nontrained designers to create online learning courseware without assistance.

KEYWORDS

Asynchronous learning: Communication between learners and instructors that does not take place simultaneously or real time. An example of asynchronous learning is when a learner engages in a self-paced, self-service learning module without communicating with another person.

Authoring tool: A software application used by non-programmers to assemble digital media files into displays, presentations, and interactive exercises.

Automated instructional design (AID): Leveraging technology to automate the instructional design as part of the process of learning content development.

Dynamic pretest: A test delivered online that can literally adapt courseware to learner deficiencies identified through online testing and scale courseware to match specific learner needs.

Groupware authoring: Concurrent process of creating learning content in a team environment with multiple authors who often play different roles, such as writers, graphic artists, or instructional designers.

Learning content management system (LCMS): A multi-user software application in which learning developers can create, store, reuse, manage, and deliver digital learning content from a central object repository (database).

Novice developer: A person who has little or no training in instructional design yet has the responsibility of creating learning content using authoring tools.

Soft-skills simulation: Simulations designed for the specific purpose of teaching interpersonal skills such as leadership, coaching, facilitating, and so on.

Storyboard: A document that details and specifies on-screen text, narrative scripts, and interaction in a paper-based format before it is converted into an online course.

INTRODUCTION

What if William Shakespeare would have had access to modern word-processing software? Would it have made him a better writer? This answer is likely an emphatic no. Taking the analogy one step further, what if the vendor who produced the word-processing software added an embedded wizard to assist Shakespeare in creating sonnets with perfect iambic pentameter? How much impact might this wizard have had on improving the quality and quantity of his work? Finally, what if a vendor touted new software that encapsulates the genius of Shakespeare such that anyone (even novice writers) who uses it could write classic material with the same style and effectiveness of Shakespeare himself? Like most people, you would likely be highly skeptical.

Seems like a series of ridiculous scenarios; yet, believe it or not, similar arguments still rage on about the role of software tools used by instructional designers to design and develop online learning courseware. In addition, many organizations have heightened expectations that relatively novice, nontechnical developers, and even subject-matter experts (SMEs) should be able to use such tools to create significant amounts of both traditional and online learning. In many cases, the developers have little or no instructional design training (Nantel and Vipond, 2006), relying instead on design and development tools to provide guidance throughout the process.

Software tools unquestionably play a valuable role in the process of instructional development, but what levels of instructional design support can we expect to come from the tools, and what part must still come from those who use the tools? The purpose of this chapter is to explore what tools instructional designers and novice developers most often use to create online learning from both design and development perspectives, what types of instructional design support can be found in different types of development tools, and how these tools may (or may not) impact the ability to meet instructional needs. This chapter focuses on four different tool classifications:

• Standard authoring tools
• Front-end design and automated instructional design (AID) tools
• Learning-activity, focused design tools, such as simulations and games
• Team development/publishing tools, such as learning content management systems (LCMSs) and groupware authoring tools
STANDARD AUTHORING TOOLS

Times have changed for instructional designers who work on online learning courses. In the early days of computer-based training (CBT) and through the early transition to Web-based delivery of asynchronous, self-paced learning, instructional development teams were most often organized with clear delineation between those who designed the instruction and those who programmed or authored course material. Instructional designers were responsible, primarily, for creating the specifications, such as design documents, scripts, storyboards, etc., while authoring specialist and programmers focused almost exclusively on assembling online learning materials to match the specifications. Over time, however, the lines between these roles continued to blur, with increased pressure on instructional designers to assume a more active role in the authoring stages of development.

To illustrate current demand for required authoring skills, a random sampling of instructional design job postings was taken from monster.com, Yahoo! hotjobs.com, and CareerBuilder.com from April to May 2006. Of the postings studied, 68% specifically listed the ability to use authoring tools as a requirement for the job, in most cases naming specific authoring tools, such as Adobe Flash, Dreamweaver, Authorware, and ToolBook (see Figure 50.1). The postings came from a wide array of organizations, both large and small, including The Johns Hopkins University, Becton Dickinson, OfficeMax, Fidelity Investments, Walgreen’s, Sears, The Hartford, Merck, and the Federal Reserve Bank. In another study, organizations that create custom courseware for internal training reported on the tools they use to create self-paced learning materials. Table 50.1 shows the top ten authoring tools used most frequently, listed in order of the highest percentage of use (Chapman, 2006b). Many of these tools have existed in one form or another for over a decade, including Flash, Dreamweaver, Authorware, Director, Lectora, ToolBook, and Quest. Standard authoring tools really have not changed much since the early days of technology-based training (Foshay and Preese, 2005), even predating the movement from hard-drive-delivered and CD-ROM courses to Web-based delivery.

Authoring tools were originally designed for the purpose of allowing nonprogrammers to easily assemble media objects and preconstructed scripting code to build instructional learning applications. Most standard authoring tools use the metaphor of a blank page, allowing authors (users of authoring tools) as much flexibility as possible to create visually appealing layouts and designs while also providing a scripting language for further extensibility in the creation of complex interactions. For the most part, authoring tools operated as advertised, although they still often required a steep learning curve in order to take advantage of their features (Merrill, 1997). In many ways,

<table>
<thead>
<tr>
<th>Authoring Tool</th>
<th>Vendor</th>
<th>Companies Using Tool (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash</td>
<td>Adobe</td>
<td>92</td>
</tr>
<tr>
<td>Dreamweaver</td>
<td>Adobe</td>
<td>67</td>
</tr>
<tr>
<td>Authorware</td>
<td>Adobe</td>
<td>34</td>
</tr>
<tr>
<td>Director</td>
<td>Adobe</td>
<td>27</td>
</tr>
<tr>
<td>Lectora Publisher</td>
<td>Trivantis</td>
<td>21</td>
</tr>
<tr>
<td>Captivate</td>
<td>Adobe</td>
<td>18</td>
</tr>
<tr>
<td>ToolBook</td>
<td>SumTotal</td>
<td>18</td>
</tr>
<tr>
<td>Breeze</td>
<td>Adobe</td>
<td>6</td>
</tr>
<tr>
<td>Total LCMS</td>
<td>SumTotal</td>
<td>4</td>
</tr>
<tr>
<td>Quest</td>
<td>Allen Communication</td>
<td>4</td>
</tr>
</tbody>
</table>
constructing courses with just an authoring tool is much like building an entire house stick by stick. Because of the open nature of authoring tools and the blank page metaphor, originally there was little or no instructional design guidance to assist novice developers in creating instructionally sound courseware.

The next evolutionary stage of authoring tools was the introduction of templates, designed for two primary purposes: (1) to facilitate rapid development of course content, and (2) to provide additional instructional design guidance for novice developers. As an example of the types of templates found in authoring tools, a set of templates named CourseBuilder was created as an add-on for Macromedia’s Dreamweaver, which was initially designed for standard Webpage development and not for the creation of learning material. The learning templates are accessed from the menu bar as a wizard to walk novice developers through the process of creating complex learning interactions. The templates can be used to create a variety of test question formats, discovery exercises, and sophisticated drag-and-drop exercises (see Figure 50.2). The templates have built-in, automatic controls to record scoring information, answer judging, feedback, and remediation branching based on learner selection.

Templates also serve another valuable role in providing examples of good design and ideas for different types of instructional interventions; for example, Articulate Presenter is a popular tool for converting PowerPoint presentations into e-learning courses. To expand on its core technology, Articulate released the Rapid E-Learning PowerPoint Template Kit, which includes dozens of professionally designed templates, including screen-layout templates and, even more importantly, examples of how the template may be applied to instruction (see Figure 50.3).

Although templates provide much needed additional design guidance to authors, desktop authoring tools still lack the core functionality to fully ensure adherence to sound instructional design principles, especially in the early stages of front-end analysis, objective writing, and designing appropriate interactions across an entire course. Bell (1998) concluded that authoring tools support many possible instructional applications, some of which may be well executed and some poorly executed, but none will be created with very much guidance from the tool.

Authoring tools, however, do play an important role in the overall instructional development process, as evidenced by the demand for authoring tool experience among posted instructional design job descriptions. Instructional designers, who also may use authoring tools, must apply their own design expertise in the early stages of instructional development, then use the authoring tools to carry out the design according to specifications.
Figure 50.3 Articulate instructional design template used for providing classification examples across multiple domains.

Figure 50.4 Typical level of instructional design support offered by standard authoring tools.

Figure 50.4 illustrates the level of instructional design support one might expect from standard authoring tools. Most authoring tools provide little to no guidance in the areas of needs analysis, objective writing, high-level (course) design, script/storyboarding, and instructional design guidance; instead, they are more appropriately used during the development phases to create interaction, which provides feedback and opportunities for learners to control their own learning experience (learner control), rapid development of online course materials (as opposed to programming from scratch), and support for the creation of automated assessment. By understanding that standard authoring tools provide limited instructional design guidance, organizations can correctly position their use appropriately in the broader context of instructional development process by applying other methods and resources to the front-end design work. This discussion also suggests that standard authoring tools may not be best suited for novice development of learning content when expectations exist that the tool will provide guidance and structure through development.
FRONT-END DESIGN AND AUTOMATED INSTRUCTIONAL DESIGN TOOLS

Approaching the design and development task from the opposite end of the spectrum is a set of specialized software applications that provide guidance through the front-end of instructional development. None of these tools has achieved the same mass adoption rate experienced by standard authoring tools. Some were initially created as experimental projects, some as in-house development toolsets supporting custom development of CBT and online courseware, and still others simply as performance support tools designed to fill in the gap left by standard authoring tools. These applications also range significantly in scope of functionality and purpose. Some are designed to create high-level instructional design documents, while others span the entire development process from front-end design through the production phases of development.

Holistic systems that also provide production of courseware are often referred to as automated instructional design (AID) tools. This moniker is what sets them apart from other design tools (Gros and Spector, 1994). As a classification of design and development tools, this category provides the most instructional design guidance targeting development by novice designers and subject matter experts and others responsible for developing instruction (Muraida and Spector, 1993). Information about the frequency of use of front-end design and AID solutions is difficult to obtain, because most studies of instructional design and development tools focus on commercial products. Table 50.2 identifies some of the most high-profile projects and solutions in this area. The common thread among these tools is that they all serve a useful purpose for environments where instructional design expertise may be lacking or in short supply and for situations where subject matter experts and others are primarily responsible for developing instruction (Muraida and Spector, 1993). Figure 50.5 illustrates where front-end design and AID solutions provide the most support for instructional development, with strong emphasis on needs analysis, objective writing, high-level design, script/storyboarding, and ISD guidance.

A simple subclassification of front-end design and AID tools was presented by Kasowitz (1998): (1) those systems that focus on cognitive aspects of instructional design—in other words, systems that prescribe instructional interventions based on best practice theories (e.g., IDExpert, ADG), and (2) those systems that provide support and guidance for the procedural steps of instructional design (e.g., AIM II, The Designer’s Edge, Gaida, DesignWare, Coursewriter). As an example, IDExpert (ID2 Research Group, led by Dr. M. David Merrill) was based on instructional transaction theory (Cline and Merrill, 1995). Briefly, IDExpert included the intelligence to bring together elements of knowledge representation, best practice instructional strategies, and instructional design prescriptions, automatically producing all of the necessary interactions for a student to acquire a particular knowledge or skill (Merrill, 1999). IDExpert users followed

Figure 50.5 Coverage of design guidance supported by automated instructional design tools (AIDs).
steps provided by the system to enter appropriate content, and the system automatically generated and delivered the appropriate interactions until mastery was achieved by learners.

By contrast, The Designer’s Edge (Allen Communication) uses a performance support approach by emulating the analyze, design, develop, implement, and evaluate (ADDIE) model of instructional design as its primary interface (see Figure 50.6). Instead of prescribing specific design methodology, the system walks novice and expert designers through the common steps of instructional design, such as conducting a needs analysis, creating an audience profile, using an embedded wizard to create appropriately formatted

---

**TABLE 50.2**

<table>
<thead>
<tr>
<th>Design or AID Tool</th>
<th>Reason Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM II</td>
<td>Created for the Navy to promote uniform instructional design practices and as a repository for course design specifications</td>
</tr>
<tr>
<td>The Designer’s Edge (Allen Communication)</td>
<td>Created as a commercial application focusing on front-end analysis, objective writing, design specifications, storyboarding, and evaluation</td>
</tr>
<tr>
<td>GAIDA</td>
<td>Created by Armstrong Laboratories at Brooks Air Force Base for the purpose of providing instructional design guidance and examples during the front end of instructional development; based on Gagné’s nine events of instruction</td>
</tr>
<tr>
<td>IDEXpert</td>
<td>Created at Utah State University through grants and sponsorships to put into practice the theory of automated instructional design</td>
</tr>
<tr>
<td>ADG (acronym for “didactic engineering workbench” in French)</td>
<td>Created as joint project of research and academic institutions as an intelligent support system for course design</td>
</tr>
<tr>
<td>DesignWare (Langevin)</td>
<td>Created as a commercial, support tool to accompany Langevin’s train-the-trainer workshops; focuses on guidance and organization course design specifications</td>
</tr>
<tr>
<td>CourseWriter (Darryl Sink and Associates)</td>
<td>Created as a commercial tool focused on, for example, front-end design, objective writing, and course organization</td>
</tr>
</tbody>
</table>

---

**Figure 50.6** The Designer’s Edge interface based on the standard ADDIE model.
performance objectives, creating a master course map, and providing extensive tools for creating scripts and storyboards to be used by multimedia programmers when creating online learning.

To provide guidance beyond the ADDIE model interface, the system also serves up context-sensitive instructional design advice and guidance from leaders in the field, linked to tools that can be used to complete the task. Although Allen Communication has added tools to transmit the learning content from the storyboarding tool directly into standard authoring tools such as Authorware, ToolBook, and Dreamweaver, the primary focus of the system is on the front-end design tasks of instructional development.

Among the examples of successful front-end design developed by utilizing AID tools is GAIDA (Guided Approach to Instructional Design Advising), based on Gagné’s nine events of instruction (Gagné, 1991). GAIDA was put through a series of tests where novices created course designs with only guidance from the support tool. The results concluded that the tool was “generally effective” in creating instructional sound learning content (Spector and Whitehead, 1994). The theory behind the tool, according to Gagné himself, was that it would “provide novices with easily understood, high-level guidance along with completely worked examples, and they will perform as if they were advanced apprentices” (Gagné, 1991). The tool is still available for a nominal fee from Cyberlearning (www.cyberlearning.com). The name of the product has been changed to GUIDE.

AIM (Authoring Instructional Materials) II has also proven to be a successful endeavor with the U.S. Navy; in fact, it is now the largest single database of training in the Navy (Arnone, 2001). In addition to providing instructional design guidance, it also serves as a primary content management system and monitors the workflow of instructional development projects. The Navy also uses the system to maintain design consistency for internal development and managing outsource course developers (Johnson, 2005).

With the promise and proven success (in part) of front-end design and AID solutions, it is a wonder why these systems have not garnered more widespread adoption, especially as more and more organizations look to novice developers and subject matter experts as course designers. This will continue to be an area of further research and study, and these early precursor projects will help set the stage for the next generation of front-end design and AID systems.

**LEARNING-ACTIVITY, FOCUSED DESIGN TOOLS (SIMULATIONS, GAMES)**

Another classification of tools used by instructional designers consists of simulation and gaming tools, designed for a precise purpose and to achieve very specific instructional outcomes. From the very beginning of computer-facilitated instruction, simulations and games have been an important instructional intervention to support experiential learning and to reinforce higher level, cognitive learning methodologies. Early simulations and games were often custom programmed at the coding level or assembled through sophisticated use of standard authoring tools. Of course, simulations have been around for many years in a variety of forms, including large-scale flight simulators, role-playing simulations in the classroom, computer-delivered simulations, and, most recently, high-fidelity simulations that can be delivered through cyberspace.

In a recent study, organizations indicated that they would like to use simulations and games as part of a blended curriculum, but they feel that the barriers to entry (such as high development costs and difficulty of creating simulations on their own) are simply too prohibitive (Chapman, 2005). In fact, the study found that the average development time required to create 1 hour of simulation was 750 hours (ranging up to 1300:1), compared to an average ratio of 220:1 for standard e-learning courses. This explains why simulations and games are currently underutilized in online learning courses; however, this desire to utilize simulations has also resulted in an explosion of a whole new line of instructional design and development tools that focus on the development of simulations and games. The following taxonomy lists the major groupings of tools found on the market today (representing over 106 simulation tool providers):

- Software simulation
- Soft-skills simulations
- Role-playing (e.g., conversation with on-screen characters)
- Business skills (e.g., making decisions to play out a business scenario such as running a mock business)
- Business modeling/analytical (e.g., setting variable conditions and observing the outcome based on business rules; learning how to interpret data)
- Story-problem/scenario-based (e.g., setting up story problems and having learners make decisions to solve the problems)
• Sales process simulator (simulating mock sales scenarios)
• Hard skills/technical
• Troubleshooting/diagnostic (making decisions and observing the outcome of each action)
• Procedural walk-through (learner performs steps in a procedure)
• Simulating physical systems (e.g., simulating pieces of equipment or other objects, setting up a computer network)
• Simulating concepts (e.g., simulating a schematic diagram, simulating how weather patterns work using a diagram)
• Emergency response simulations (e.g., performing actions as a result of an emergency)
• Virtual worlds/spatial relationships (e.g., flight simulators or simulating an office environment, cockpit, or factory)

Some of the leading tools in each of these sectors are shown in Table 50.3.

Software Simulation Tools

Software simulation tools allow novice developers to record screen interactions while walking through a procedure. The tool remembers each mouse click and keystroke for the purpose of later creating interactive simulations—for example, registering the coordinates and creating a click area for each mouse click. Simulation authors must simply clean up the recording and add feedback and remediation to complete the simulation. This is considerably quicker than using standard authoring tools to (1) capture screens, (2) manually add touch areas and text input fields, (3) create prompts and instructions for the learners, (4) wire each click or correct key entry into a branching pattern, etc. Software simulation tools have significantly reduced development time, while preserving a high degree of fidelity and interactivity in the learning event.

Soft-Skills Simulation Tools

Soft-skills tools (Table 50.4) are usually based on a template model for quickly creating specific types of interactions such as developing a dialog, choosing on-screen characters, and creating a role-play simulation with graded responses and comparisons to expert paths through the simulation (how they may have handled the situation). The difficulty for makers of simulation tools is that they must try to provide the flexibility necessary to vary the simulation for different purposes while keeping focused on delivering high-fidelity, workplace-relevant scenarios created in a very short amount of time. Figure 50.7 shows an example of a sophisticated simulation in which the learner makes decisions about how society may potential reverse the effects of global warming. The simulation, from Forio, is capable of simulating elapsed time as the learner may run the scenario over months or even years (compressed to an hour or less online).

Hard-Skills/Technical Simulation Tools

The hard-skills/technical simulations category (Table 50.5) covers a broad range of skill areas. Think of systems in this category as being capable of modeling physical systems, such as a piece of machinery or an electronic measurement system. This category also covers task simulators designed to teach and measure a learner’s ability to follow steps in a procedure with various levels of guidance. A good example of how this might work is the flight simulators often used in the aviation industry. Many airlines have figured out how to pass performance data from sessions in a flight simulator back to a central learning management system, which keeps performance records for classroom-based learning and e-learning courses.
Games play a unique role in deploying an enterprise-wide learning strategy (Table 50.6). Although not primarily designed to deliver an entire training or teaching job alone (Kirk and Belovics, 2004), games provide a unique level of motivation in learning areas such as classification (matching games), reinforcing factual information (question and answer games), and rote learning (memorization games). In addition, as the new gaming generation enters the workforce, lecture-based training will likely be less effective (Kirk and Belovics, 2004). What makes simulation and gaming development tools unique is that instructional design guidance is not often obvious through the interface; rather, guidance comes from the intelligence and flexibility of the simulation or game designed to achieve a specific learning outcome. The primary instructional benefits of using such tools is the depth of interaction and rich feedback available for learners, achievable through relative rapid development processes and minimizing the authoring savvy required on the part of instructional developers (see Figure 50.8).

**TEAM DEVELOPMENT/PUBLISHING TOOLS**

Since the beginning of 2000, a new classification of instructional development tools has evolved that is beginning to gain traction as commercially viable options for developing online learning. Many of the most popular traditional authoring tools available today are desktop applications, meaning that they typically can be used by only one instructional developer at a time. However, newer tools, such as those designed for the web or for use on mobile devices, allow multiple users to participate concurrently in the learning experience.
time to create interaction. In contrast, this new classification of tools is based on groupware projects based on a publishing model for online learning development, using multiple, concurrent content contributors, each working on learning content and interactive exercises that are part of a larger course or curriculum. To illustrate the point, consider the following question: Could the publishers of *The Los Angeles Times* use Microsoft Word to lay out and organize each daily newspaper? The answer is “yes.” How efficiently could it be produced, though? The answer is “not very efficiently.”

These groupware publishing systems are called *learning content management systems* (LCMSs). By definition, an LCMS is a multidevolved environment where developers can create, store, reuse, manage, and deliver learning content from a central object repository (Chapman, 2006a). Although many LCMS solutions have built-in authoring capabilities, most are also designed to assemble individual learning activities that may have been created using a variety of traditional authoring tools and storing the learning events as learning objects. A learning object is a reusable learning activity that can be (1) metadata tagged for easy retrieval, (2) standards that can communicate with other learning technologies, or (3) objects combined or clustered with other learning objects to create new, derivative learning structures such as lessons, units, or entire courses. Table 50.7 lists some of the most frequently used commercial LCMS solutions to date (Chapman, 2006b).

Table 50.7

<table>
<thead>
<tr>
<th>Product</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolution</td>
<td>Outstart</td>
</tr>
<tr>
<td>TotalLCMS</td>
<td>Sum/Total</td>
</tr>
<tr>
<td>ForceTen</td>
<td>Eedo</td>
</tr>
<tr>
<td>Saba LCMS</td>
<td>Saba</td>
</tr>
<tr>
<td>Learn.com</td>
<td>Learn.com</td>
</tr>
<tr>
<td>Generation 21</td>
<td>Generation 21</td>
</tr>
<tr>
<td>TopClass</td>
<td>WBT Systems</td>
</tr>
</tbody>
</table>

Beyond the rapid development capabilities offered by these systems, LCMS tools often also include a wide range of embedded instructional design guidance, at times bordering on a hybrid between standard authoring and front-end design and automated instructional design (AID) tools. Following are some of the areas of instructional design guidance that can be found inside commercial LCMS solutions.

**Interactivity Templates**

Learning content management system tools offer even more extensive capabilities for using and reusing design templates. One example would be a gaming template that is driven by a bank of test questions. The template provides the interface, look and feel, game show host, game boards, etc. Because all learning content in an LCMS is stored in a database, the test questions can be randomly and automatically extracted by the database each time a new version of the game is played; hence, the same template could yield thousands of variations of the same gaming experience.
Bryan L. Chapman

Dynamic Pretesting

Most LCMS products use a dynamic pretesting model, meaning that individual test questions can be linked with learning content (stored as learning objects). Based on learner performance in the test, the LCMS can dynamically create a new version of the course based on demonstrated performance in the test, so learners do not receive instruction on content they have already mastered. When used correctly, dynamic pretest can dramatically shorten course contact time.

Learning Objects

Although most references to learning objects focus on their reusability, there is a more significant instructional tie-in. Learning objects are based on a single, enabling learning objective, thus providing focus and clarity on teaching and assessing learner performance against their mastery of the objective. Learning objects—with a one-to-one relationship with enabling objective—can be clustered with other enabling objectives to create new paths for working toward a broader terminal objective (the purpose of a lesson or entire course). Learning object design helps focus instructional designers on their ultimate goal.

Storyboard Specifications

In the instructional development process, storyboards represent a detailed design specification for the course as an elaboration of high-level design documents. LCMS technology provides methods that allow learning objects to be based on objectives and then designed into delivery specifications via storyboarding in support of audio/video production, message design, and keeping track of complex details of interactivity. Figure 50.9 shows an example of an LCMS-generated storyboard interface for QMIND.

Links between Assessment and Competencies

The makeup of LCMS technology makes it possible to create sophisticated links between test questions and competencies (skills, knowledge, and attitude).
Programming numerous connections between learning performance and competencies is possible using standard authoring tools, but the complexity makes it a barrier to measuring learning performance in a meaningful way for novice developers. In general, LCMSs provide less guidance than front-end and automated design tools (see Figure 50.10). The structure of most systems is based on the notion that front-end analysis and high-level design have already previously been completed and that the purpose of the LCMS is to support the remaining development and delivery process. They do encourage developmental practices but still come up short in supporting the full range of instructional design guidance across the entire spectrum; however, of all the commercially viable learning technologies, LCMS technology appears to be evolving toward a platform that may show promise in providing both instructional design guidance and development system.

**CONCLUSIONS**

At present, myriad online learning development tools cover a broad spectrum of choice and flexibility; yet, the sheer number of these tools, plus new emerging technologies such as wikis, blogs, and podcasts, leaves organizations in a state of confusion when attempting to develop a cohesive authoring and development strategy. Although instructional design guidance, in some form or another, is now embedded inside several classifications of development tools, considerable innovation and research are still required to create holistic systems that negate or minimize the need for instructional design expertise, allowing nontrained designers to create online learning courseware without assistance.

Perhaps software developers will never create a wizard that allows writers to fully write like Shakespeare, and the same may be true for intelligent, instructional design software, but that does not mean that instructional design guidance and principles should not be embedded in the evolution of existing content development tools or future technologies. In the meantime, it is incumbent on every organization that creates online learning to find the appropriate balance and mix between internal expertise (people resources) and available tools and technology for designing instruction.

Great appreciation is extended to those who continually push the envelope and introduce instructional design savvy into the tools of the trade. They have laid a great foundation for future innovators to draw upon to create new methods, new tools, and new solutions, as long as they keep their vision clearly focused on the ultimate goal—namely, the design and creation of effective learning experience that result in optimal knowledge transfer.

**Figure 50.10** Instructional design support offered by learning content management systems (LCMSs).
REFERENCES


* Indicates a core reference.