Choosing Authoring Tools

Advanced Distributed Learning (ADL) Initiative

Peter Berking

7 July 2016
Version 9.5.7

Table of Contents

NOTE: Vendor citations or descriptions in this paper are for illustrative purposes and do not constitute an endorsement by ADL. All listings of vendors and products are in alphabetical order unless otherwise noted.

1. Purpose and scope of this paper

The purpose of this paper is to help those involved in the process of choosing authoring tools to make an informed decision. The paper presents a range of considerations for choosing tools, whether as an enterprise-wide acquisition or a single user purchase, and includes a sampling of current tools categorized according to the kind of product they are intended to produce.

This paper does not contain a comprehensive survey of available tools on the market, nor does it contain a comparative rating or evaluation of products, and should not be construed as having such. For more in-depth information about tools and their features, see the references in 10 *References cited in this paper*, or consult the vendors. ADL presents this paper merely as a guide to the issues, opportunities, and processes that are typically considered in choosing authoring tools.

ADL has titled this paper "Choosing Authoring Tools" rather than "Choosing an Authoring Tool," since there is usually a need to select more than one product. Rarely will one tool meet all the production needs of an organization or developer. Most developers use a combination of tools, even to produce a single eLearning product; using a combination of tools that are each optimized to produce particular components of the product can increase production efficiencies dramatically. Additionally, you may find it impossible to create the variety of eLearning product types your organization requires with a single tool. A survey of authoring tools reported that respondents use an average of 3.35 tools (Shank & Ganci, 2013).

In line with our mission to promote reusability and interoperability in eLearning, ADL recommends authoring tools with built-in features that allow creating SCORM®-conformant eLearning. Creating eLearning that is not reusable or interoperable can be a significant business risk, since you may not be able to run your content in more than one LMS, and you may needlessly develop already-existing content. You can find SCORM considerations for authoring tools in 4.10.1. *SCORM*.

2. Overview

2.1. What is an authoring tool?

Authoring tools are software applications used to develop eLearning products. "eLearning" is defined in the broadest sense as "… referring to the delivery of training or educational programs using technology to enable people to learn anytime and anywhere." (ID Guru, 2011). Also, "...eLearning can be both synchronous and asynchronous, on standalone computers or devices, or on Web-based networks." (TrainingIndustry.com, 2015)

Authoring tools generally include the capabilities to create, edit, review, test, and configure eLearning. These tools support learning, education, and training by enabling using distributed eLearning that is cost-efficient to produce, and that facilitates incorporating effective learning strategies and delivery technologies into the eLearning.

Authoring tools range from advanced software to create a wide array of sophisticated applications (not limited to eLearning) to simple tools that convert instructional PowerPoint® slides to web pages. In this regard, it is important to understand that some software tools used as authoring tools are not necessarily designed for the creation of eLearning specifically; they can be open-ended, multi-purpose tools designed

to create, for instance, any kind of web page/site. But when developers use them to create eLearning, they are referred to as authoring tools.

Vendors build some authoring tools into systems that perform broader functions; this is the case with learning content management systems (LCMSs). *See* 3.2. *Learning content management systems (LCMSs)* for more details. In many LCMSs, you can decouple the authoring tool component and use it as a separate application to develop and output eLearning without relying on the other components of the system.

As described in 1. *Purpose and scope of this paper*, developers rarely use authoring tools in isolation; in fact, most developers use more than one software tool during the production process, and a substantial number use four or more. Even when using a combination of tools, however, a developer generally uses one primary tool to create the base screens and assemble them into an eLearning product. These tools are distinguished from auxiliary software tools (for instance, Adobe Photoshop®) that are not authoring tools but may be used in support of or in conjunction with those tools. This paper includes a discussion of auxiliary tools.

Authoring tools discussed in this paper refer to web-based eLearning (or web-based training, WBT); CD-ROM or DVD-based eLearning has largely disappeared due to the establishment of enterprise intranets and extranets, and the distribution efficiencies of using web-based delivery. However, many authoring tools offer disc-based delivery as an output option in order to support environments where bandwidth is limited or non-existent.

2.2. Why use authoring tools?

Authoring tools (as opposed to writing code or script directly in a programming editor) reduce technical overhead; they generally use WYSIWYG ("what you see is what you get") interfaces allowing users to easily manipulate and configure eLearning assets, using familiar visual metaphors. Thus, programming editors that facilitate writing application code like C++ or script languages like JavaScript are not truly authoring tools. Developers can indeed use them to author eLearning content, but they are not designed to reduce the technical overhead of knowing the programming or scripting language. Furthermore, most training organizations do not have the advanced (and expensive) programming skill sets in their development staff to program eLearning applications using only programming languages or scripts, and they do not have the infrastructure to support code-based traditional software application development.

Primarily, authoring tools serve to reduce the skill set requirements for the authoring process, in some cases to a level where an untrained user can start using a tool and producing screens within minutes.

Secondarily, most authoring tools base a major part of their value-add proposition on automating time-consuming tasks, optimizing workflows, and generally offering a more streamlined and efficient approach to the authoring process, which can be very time consuming.

2.3. Why is the choice of tools so important?

Choosing eLearning authoring tools is one of the most crucial decisions any training organization, project, or developer can make. Authoring tools are designed for particular styles of learning, delivery platforms, file formats, eLearning standards, and production workflows. If your organization chooses a tool or set of tools that is not optimized for your needs, you could waste a lot of time and money creating eLearning that does not function correctly within your training infrastructure or that is instructionally ineffective.

Another critical factor in choosing tools—one that can make or break an organization's training budget with costly conversions—is durability. This relates to whether the tools will have longevity in the marketplace such that they continue to be available and supported, allowing source files to be opened and

edited in future versions of the application. It also relates to whether the tools will, in the future, produce output formats supported by browser versions and browser plug-in updates.

2.4. Should my organization mandate use of standard tools?

Many organizations wonder whether they should mandate adopting a particular set of tools as a required standard across their organization. This has many advantages, among them:

- Reducing costs through purchase of group or enterprise licenses that lower the per end-user cost
- Providing for economies of scale in training to use the tools, help desk support, configuration management, etc.
- Making enforcing uniform eLearning product standards easier through dissemination of application source file templates

The most important consideration in whether to standardize on tools, however, is the variability in types of training your organization produces. As stated above, authoring tools are optimized for particular types of training or IT environments. Mandating use of a single tool set as the organizational standard can effectively amount to enforcing one style or type of training across the organization, which may be counter to the organization's (or even single project's) needs. More and more nowadays, training programs incorporate disparate elements in a blended or hybrid solution.

For instance, you may decide that the best way to teach some skills in a course is through asynchronous eLearning, while you may decide to teach other skills in the same training course through a synchronous virtual classroom environment. The choice of authoring tools probably will not be the same for both. You must take this into account in developing the tool standard specifications, when tools are standardized throughout the organization. The standard must address each type of learning, file output type, etc., with a standard tool set specification for each type. Seldom will one tool set suffice to cover all aspects of the authoring process or meet all needs for the various types of training produced by the organization.

Before specifying tool standards, ADL recommends that you standardize the requirements for the eLearning products themselves, using style guides and other policy documents. This includes such things as delivery device, look and feel, interface functionality, file formats, course elements, and assessment design. This will drive and clarify the choice of tools.

3. Categories and examples of authoring tools

Authoring tools run a wide gamut. This section outlines the major categories and subcategories of available tools. These categories are key to choosing an authoring tool, since they set the stage for allowing you to align your eLearning product requirements to tool types and characteristics. It is important to note that these categories are not mutually exclusive. Many tools have elements that qualify them for two or more categories. However, most tools can be assigned to one category as its primary intended use or design thrust.

The following is an outline and description of the types of authoring tools, with examples. The websites listed for each provide feature sets and further details on each tool. Note that some tools appear in more than one category, as they fulfill multiple purposes.

Tools that are open source, GOTS (government off-the-shelf), or freeware are indicated. All other examples are COTS (commercial off-the-shelf) products. For more information on open source, freeware, or GOTS, see 4.3. *Open source, freeware, and GOTS solutions.*

Note: the lists of examples are not comprehensive, nor do they represent an endorsement of particular products. They are based on ADL's knowledge and ongoing research as of the date of this document. "Section 9: For further reference" lists web sites that may provide more comprehensive and updated information about specific tools.

3.1. Self-contained authoring environments

These applications enable building entire eLearning courses using capabilities within the authoring tool; they do not rely on externally created documents (except for media assets and possibly databases). These generally incorporate WYSIWYG features for screen layout and design, and use an object-oriented approach for structuring course elements and activities.

3.1.1. Website development tools

These are open-ended tools for website design; they can be used for any type of website or web pages, including eLearning. Once your organization has developed templates and established workflows, these open-ended tools can work well for authoring eLearning. All create output in standard eLearning web formats using HTML, CSS, and Javascript. Examples are:

- Dreamweaver®
  http://www.adobe.com/products/dreamweaver/
- Visual Studio 2012®
  http://www.microsoft.com/visualstudio/eng/team-foundation-service

3.1.2. Rapid Application Development (RAD) tools

These are open-ended tools for designing robust interactive applications (usually for web delivery). They produce binary runtime files that are executed by a player or plug-in. Examples include:

- Flash®
  http://www.adobe.com/products/flash/
- Flex [open source]
  http://www.adobe.com/products/flex/

3.1.3. eLearning development tools

These tools are specifically designed to produce eLearning, generally in one or two output file format options. These systems are what training professionals are most commonly referring to when using the term "authoring tools." The system architecture often relies heavily on templates and "skins" to maximize production efficiencies. In some cases, the developer cannot create templates; the vendors must create them. In addition to template-based tools, there are two other types: timeline-based tools and object-based tools.

Timeline-based tools allow authors to create a sequence of actions on a timeline. These tools tend to be more powerful in that they can natively support authoring animations and object state dependencies. These two features in combination can be used to create simulations.

Object-based tools allow authors to build content using predefined objects with highly configurable properties. Objects can include a wide variety of screen elements, such as search capability, wikis, etc.. Object-based tools can be thought of as a variation on the theme of template-based tools in the sense that the individual objects are essentially the templates. They are less constrained because these objects can be mixed and matched at a much finer-grained level than screen templates. Object-based tools are usually

more technical and complex than template-based tools, thus they require more development time and training.

The simpler, easier-to-use tools in this category are sometimes loosely referred to as "rapid eLearning development tools" due to both the speed with which authors (especially those that are not technically inclined) can learn to use the tool, and the speed of production. However, the term is generally better suited for the tools described in 3.7. *External document converter/optimizer tools*.

3.1.3.1. Cloud-based eLearning development tools

These tools are cloud applications that are installed on a cloud server and use the web browser as the application interface, as opposed to being installed on your local computer. Some of these cloud-based authoring tools require installation of a thin desktop client or a browser plug-in. The following are some of the advantages of cloud-based authoring tools:

- Allows authors to see the same content at the same time, and thus collaborate on it simultaneously. Desktop authoring tools require you to send files to other authors sequentially and track versions manually.
- Enables centralized control and enforcement of templates, standards, skins, etc.
- Enables permission/role-based production workflows.
- Updating and configuring the tool is centrally managed. Everyone is always running the latest version, since there is only one copy of the software on the cloud server for administrators to deal with. Desktop authoring tools can be a problem, if versions are not in synch and the feature sets are not the same, or, even worse, versions are so different that they don't accept files transferred between them.
- They can provide up-to-the-minute, aggregated data about usage, project progress, etc.

Examples include:

- Brainshark Learning Cloud®
  http://www.brainshark.com/solutions/learning-cloud.aspx
- CHOOSE-IT [under development by Army Research Institute for use in DoD only]
  [contact Dr. Cheryl Johnson for details: cheryl.i.johnson@us.army.mil]
- Claro®
  http://www.dominknow.com
- Course Avenue Studio®
  http://www.courseavenue.com
- D2 Interactive Multimedia Instruction Framework®
  http://www.d2teamsim.com/d2-products/DIF.html
- GoMo Learning Suite®
  http://www.gomolearning.com/
- Ilias SCORM Editor [open source]
  http://www.ilias.de
- Lectora Online®
  http://trivantis.com/products/lectora-online-authoring/
- Luminosity Studio®
  http://www.cm-group.co.uk/products/elearning-authoring-solution/

- Mohive®
  http://www.crossknowledge.com/en_GB/elearning/technologies/mohive.html
- Oppia [open source]
  https://www.oppia.org/
- RapideL®
  http://www.rapidel.com/
- Litmos Author®
  https://www.litmosauthor.com/
- Skilitix Interact®
  http://skilitics.com/interact/overview/
- SmartBuilder®
  http://www.suddenlysmart.com/smartbuilder.htm
- Udutu [freeware]
  http://www.udutu.com/
- ZEBRAZAPPS®
  https://zebrazapps.com/

3.1.3.2. Desktop-based eLearning development tools

Many vendors are moving away from desktop-based authoring applications since they cannot be used collaboratively; some are retaining desktop-based versions as an option. Desktop-based applications generally perform better than their web-based cousins, and have more features. Some desktop tools (for example, those with video editing tools) do not have web counterparts due to high minimum performance requirements. Examples include:

- Adapt [open source]
  https://community.adaptlearning.org/
- Captivate®
  http://www.adobe.com/products/captivate/
- Content Publisher®
  http://www.elicitus.com
- Course Builder [open source]
  https://code.google.com/p/course-builder/
- CourseLab® [available as a commercial product (latest version) and as freeware (earlier version)]
  http://www.courselab.com/view_doc.html?mode=home
- e-Learning Suite® [a suite of tools – the primary authoring tool is Captivate]
  http://www.adobe.com/products/elearningsuite/
- eXe [open source]
  http://exelearning.org/
- EXPERT Platform [open source – limited to government and non-profit organizations]
  for information contact Bill Bandrowski – band@ctc.com
- Expert Author®
  http://www.knowledgequest.com

- GLO Maker [open source]
  http://learning.londonmet.ac.uk/RLO-CETL/glomaker/index.html
- Impression Learning Content Framework®
  http://impressionlcf.com/
- iSpring Suite®
  http://www.ispringsolutions.com
- Learn®
  http://www.sumtotalsystems.com/enterprise/learn/
- learningMaker®
  http://www.netexlearning.com/en/learningmaker/
- Learning Suite®
  http://www.kenexa.com
- Lectora Inspire®
  http://trivantis.com/products/inspire-e-learning-software/
- MOS Solo® [free]
  http://www.moschorus.com/centre/MosPub/solo_en/index.html
- Multimedia Learning Object Authoring Tool® [freeware]
  http://www.learningtools.arts.ubc.ca/mloat.htm
- Obsidian Black®
  http://obsidianlearning.com/
- SmartBuilder®
  http://www.smartbuilder.com/product/smartbuilder
- Storyline®
  http://www.articulate.com
- Xerte [open source]
  http://www.nottingham.ac.uk/xerte/xerte.htm

3.1.4. Simulation development tools

These tools are specifically designed for developing simulations and their component animations. Some incorporate scientific data sets that allow modeling of physical phenomena to simulate the real world as closely as possible (for example, weather conditions in a flight simulator). Most Rapid Application Development (RAD) tools can create simulations as well.

3.1.4.1. System simulation development tools

These tools are optimized for systems training, producing essentially a recording of what is happening in a computer screen (often called “screencasts”). They allow easy capture and captioning of interface features with voiceover narration, additional graphics, and interaction. Examples include:

- Assima Training Suite®
  http://www.assima.net/training-suite.html
- Captivate®
  http://www.adobe.com/products/captivate/

- Camtasia Studio®
  http://www.techsmith.com
- Firefly Simulation Developer®
  http://www.mzinga.com/products/omnisocial-content/firefly-simulation-developer/

3.1.4.2. 2D simulation development tools

These tools are used to create 2D simulations. They are much simpler to learn and use and less expensive than 3D simulation development tools, but more difficult and costly than eLearning development tools. Examples include:

- SimWriter®
  http://simwriter.com/
- GoAnimate®
  http://goanimate.com/

3.1.4.3. 3D simulation development tools

These tools are used to create 3D simulations, usually that look and act like the physical world. The tools can either model the physical world using geotypical or geospatial data. Geotypical modeling renders artifacts and environments using databases of scientific data sets that predict, for example, the state of cloud cover over a location at a certain time of the year and day (not limited to the clouds' appearance, but also including physical properties such as altitude, moisture content, etc.). The clouds are then generated synthetically (as vector-based 3D models) using a library of textures and skins, and can interact with other items in the environment based on their assigned physical properties.

Geospatial modeling renders artifacts and environments using satellite imagery, archived photographs, GPS surveys, and live data feeds from sensors. This type of modeling would render the state of cloud cover over a location for a particular date and time, as it truly exists or existed. It may include their physical properties as they actually exist/existed as well.

Geotypical modeling is more flexible and better suited for most simulations, since it allows on-the-fly, dynamic changes to the physical appearance and attributes controlled by either the user or simulation itself. This permits a wide variety of "what if" scenarios. Examples of 3D simulation development tools include:

- CodeBaby Studio®
  http://codebaby.com/elearning-solutions/get-free-trial-now/?cpid=ELGad100112
- ESP® [this tool is still available, but Microsoft no longer supports it]
  http://msdn.microsoft.com/en-us/library/ff798293.aspx
- Flash Builder®
  http://www.adobe.com/products/flash-builder.html
- Flex SDK® [open source]
  http://www.adobe.com/products/flex/
- Kuda® [open source]
  http://code.google.com/p/kuda/
- SimWriter®
  http://www.simwriter.com

3.1.4.4. Video role play tools

These tools allow you to create video scenarios that learners can respond to using a computer and webcam. Learners can review and share submissions with mentors and other learners. Examples include:

- Rehearsal VRP®
  http://www.rehearsalvrp.com

3.1.4.5. Transmedia story-based tools

These tools allow you to create immersive learning scenarios (usually involving real world actions). These scenarios unfold across multiple media types and channels, sent in timed sequences to build a "story" that engages the learner. The media channels could include web, mobile, email, social media, SMS, phone calls, IoT, and others. These tools involve authoring tool and LMS elements; they are more aptly categorized as "experience managers".

Examples include:

- Conducttr®
  http://www.conducttr.com

3.1.5. Game development environments

Although you can use many RAD and simulation tools to create game-based learning applications, tools in this category are specific to a particular game engine or game standard, or have gamification of learning as a central feature. Examples include:

- GameSalad® [optimized for producing mobile games]
  http://gamesalad.com
- GameStudio®
  http://www.3dgamestudio.com/
- Knowledge Guru [LCMS with game elements built in]
  http://www.theknowledgeguru.com/
- Torque Game Engine®
  http://www.garagegames.com/
- Truevision 3D®
  http://www.truevision3d.com/page-14-create-3d-game-development
- Unity Pro®
  http://unity3d.com
- VBS Worlds®
  http://www.vbsworlds.com/
- Visual3D®
  http://www.visual3d.net/

3.1.6. Virtual world development environments

Although you can use many RAD, simulation, and game development tools to create virtual world learning applications, these refer to those that are specific to a particular virtual world or virtual world type. Examples include:

- OpenQwaq [open source]
  https://github.com/itsmeront/openqwaq
- OpenSimulator [open source]
  http://opensimulator.org/wiki/Main_Page
- Open Wonderland [open source]
  http://openwonderland.org/
- Protosphere®
  http://www.protonmedia.com/
- Second Life®
  http://www.secondlife.com
- Vastpark®
  http://www.vastpark.com/
- Virtual World Sandbox [open source]
  http://adlnet.gov/adl-research/virtual-reality-games-simulations/virtual-world-sandbox/
- Vizard Virtual Reality Toolkit®
  http://www.worldviz.com/products/vizard
- World Visions®
  http://www.aesthetic.com/home_frame/home_frame.htm

3.1.7. Database-delivered web application systems

These tools represent the ultimate leveraging of the concept of separation of content and appearance; developers store the content (text and media assets) in a database, and apply formats to them on a presentation layer at runtime. This can be a great advantage if learning content information is volatile; you can update content simply and cleanly by replacing objects in a database through a web form. This approach can minimize course maintenance costs for clients by allowing them to make minor updates themselves rather than paying the content developer for every change.

The authoring tools rely on manipulating screen placeholders (that call objects in from the database), and provide form-based methods for configuring and populating the database. These tools require server software to deliver the eLearning. Examples include:

- ColdFusion®
  http://www.adobe.com/products/coldfusion/
- ASP.Net® [programming language built in to all Microsoft servers]
  http://msdn.microsoft.com/en-us/centrum-asp-net.aspx

3.2. Learning content management systems (LCMSs)

These applications integrate the authoring functions with content management, storage, and delivery, leveraging the advantages of integrating these functions. They also generally assemble and deliver the eLearning dynamically at runtime from a central content repository. This provides great flexibility for reuse of content and media. Users do not develop actual files during the authoring process; they assemble virtual learning objects from database and file elements, similar to the database-delivered web application systems described in section 3.1.7. *Database-delivered web application systems.* See the ADL paper *Choosing an LMS* at http://adlnet.gov/adl-assets/uploads/2016/01/ChoosingAnLMS.docx for more details on LCMSs. Examples include:

- ATutor® [open source]
  http://www.atutor.ca/atutor/index.php
- SilkRoad Learning®
  http://www.silkroad.com/hr-solutions/talent-development/silkroad-learning/
- Impression Learning Content Framework®
  http://impressionlcf.com/
- eXact Learning LCMS®
  http://www.exact-learning.com/en/products/learn-exact-suite
- Mindflash®
  http://www.mindflash.com
- MOS Chorus®
  http://www.moschorus.com/centre/MosPub/chorus_en/index.html
- SAP Enterprise Learning®
  http://www.sap.com
- Learning Essentials®
  http://www.sumtotalsystems.com
- Xyleme LCMS®
  http://www.xyleme.com/

3.3. Virtual classroom systems

Vendors design these applications specifically to create eLearning that is delivered via an online collaboration tool (usually one that is optimized for eLearning, with familiar classroom metaphors). The collaboration functionality is usually combined with the authoring functionality in one system. LMS functions are often included as well.

Developers use these systems to author synchronous or asynchronous virtual classroom training; most are capable of creating asynchronous eLearning only by virtue of the fact that the synchronous session can be recorded and played back for self-paced learning. These are not standalone systems, because they require files to be generated externally and imported (for example, PowerPoint® slides). Examples include:

- Adobe Presenter®
  http://www.adobe.com/products/presenter.html
- Blackboard Collaborate®
  http://www.blackboard.com
- Learning@Work®
  http://www.saba.com/us/apps/learning-work/
- Connect®
  http://www.adobe.com/products/acrobatconnectpro/
- GoToTraining®
  http://www.gotomeeting.com/fec/training/online_training
- OmniSocial HR and Learning Suite®
  http://www.mzinga.com/a/pdf/MzingaDS-HRSolutions.pdf

3.4. Mobile learning development tools

Most authoring tools can now deliver content to mobile devices, simply because mLearning is becoming the dominant paradigm for delivering eLearning. The tools provide this capability by using a mobile device screen template and output files that work with the mobile device operating system.

However, tools are emerging that are specifically designed for mobile learning (mLearning), for instance, providing authoring capability for audio learning content (e.g., spoken word, podcasts) along with associated interactive assessments and surveys. Other tools are optimized to provide eLearning content through the phone's web browsing capability. Responsive design is now a key feature of many of these tools. For more details, see 4.12 *Responsive design.*

Note that some of the mLearning authoring tools are designed to run only within an integrated LMS platform; stand-alone portability isn't always possible. Also, some target only one screen size (for example, the Apple iPad®). Some of them support SCORM output (for details on SCORM implementation strategies, see https://docs.google.com/viewer?a=v&pid=sites&srcid=YWRsbmV0Lmdvdnxtb2JpbGUtbGVhcm5pbmctZ3VpZGV8Z3g6MzM2ZDcyMDQ0ZjkwOTZmYw).

A key feature of the self-contained integrated platforms mentioned above is the ability to provision content while on free wireless networks to reduce the cost of downloading large files over cellular networks. This applies mainly to BYOD users. The user downloads a container or player application initially and then downloads needed content while on a wireless network so that he or she can consume the content offline later.

Examples of integrated platforms that focus more or less exclusively on authoring and delivering mLearning include:

- AppCooker® [creates prototype mockups]
  http://www.appcooker.com
- LearnCast®
  http://www.learncast.com/
- CellCast®
  http://www.onpointdigital.com
- CourseAvenue Enterprise Mobile Solution®
  http://www.courseavenue.com/onecourse-solution-for-mobile
- Evernote®
  http://www.evernote.com
- EXPERT Platform [open source – limited to government and non-profit organizations.
  For information contact Bill Bandrowski – band@ctc.com]
- StoryWorks1®
  http://storyworks1.com/
- MASLO [open source – under development]
  http://adlnet.gov/maslo/
- Mobile Study®
  http://www.mobilestudy.org/
- Mobl 21®
  http://www.emantras.com

For the names of authoring tools that support mLearning (but are not integrated platforms like the above list), refer to the tools listed in the other categories of authoring tools in this document; as stated earlier, almost all of these support mLearning authoring. A key feature to look for in these tools is HTML5 output, since that is supported "out of the box" by mobile devices. Responsive design (see 4.12 *Responsive design*) is also a key feature that qualifies a tool for supporting mLearning.

A mobile learning authoring tool comparison (published June 2015) is available from the eLearning Guild at http://www.elearningguild.com/content.cfm?selection=doc.3971 (eLearning Guild, 2015)

See 4.2. *mLearning authoring tools* for more details on issues and opportunities involved in authoring for the mobile platform.

3.5. Performance support development tools

These are tools to specifically author performance support modules. ADL defines performance support (and distinguishes it from training or eLearning) as "A means of providing workers with the information and resources they require to perform a task. Example: job aids, mobile support, and EPSS. (Instructional Design Guru, 2011). Usually performance support is designed to be provided by a system at the time of need, while the user is immersed in the task. This is called "just in time" support. There can also be "just in place" support, meaning support that is triggered by the learner being in a particular location. (ADL)

In terms of the design of authoring tools, there is a lot of overlap with eLearning authoring tools, but there are some differences, mainly less (or no) emphasis on testing and assessment.

Tools include:

- Ancile
  http://www.ancile.com/
- Inkling
  https://www.inkling.com/platform/

3.6. Social learning development tools

Some authoring tools are designed to create learning that is based on learner-generated content, peer-to-peer communication, and collaboration provided by social media tools. Use of these features in eLearning is increasing rapidly; some vendors now specifically tailor collaboration tools to support eLearning and their authoring and delivery systems. These authoring tools support publishing learning modules that include such formats as:

- Wikis (for example, Wikipedia®)
- Social networking (for example, Facebook®)
- Blogs (for example, Blogger®)
- Micro-blogs (for example, Twitter®)
- Social bookmarking (for example, Delicious®)
- Social news (for example, Digg®)
- Picture sharing (for example, Flickr®)
- Video sharing (for example, YouTube®)
- Communities of practice (CoPs)
- Expert exchanges (for example, Experts-Exchange.com®)

Examples of tools include:

- Bizlibrary Community®
  http://www.bizlibrary.com
- Bloomfire®
  http://www.bloomfire.com
- Composica®
  http://www.composica.com
- Engage®
  http://www.blackboard.com/Platforms/Engage/Overview.aspx

3.7. External document converter/optimizer tools

These applications usually offer limited ability to develop eLearning from scratch; they are primarily designed to import and convert external documents (usually PowerPoint® and Word® documents) to web-based eLearning (in HTML5 or Flash® format usually). Often these external documents are legacy ILT (instructor-led training) files (student guides and presentation slides, for example) that need to be converted to eLearning.

This category of tools includes what is known as "rapid eLearning development tools." See 4.1. *Rapid eLearning authoring tools* for more information.

If you are using PowerPoint as the starting point for your content, you may not need to convert using one of these tools. PowerPoint alone can be used to produce traditional asynchronous eLearning with the look, feel, and functionality of eLearning developed in other authoring tools. This will require that the information on slides be elaborated so that the slides are self-sufficient for standalone eLearning delivery (vs rendered in abbreviated bullet points for live presentation delivery). This approach leverages some of the lesser-known abilities of PowerPoint to create clickable objects, and multiplies the advantages mentioned above of the rapid eLearning approach (since you are eliminating the conversion and optimization that would be necessary using one of those tools). However, using PowerPoint as your delivery file format has some constraints which need to be carefully considered.

The ADL white paper *Authoring and Delivering e-Learning Using PowerPoint Files* describes considerations and procedures for using PowerPoint files as your eLearning tool and delivery file format. Contact the author at peter.berking.ctr@adlnet.gov for a copy.

3.7.1. Web-based external document converter/optimizer tools

These web-based tools offer the same advantages over desktop tools described in 3.1.3. *eLearning development tools*—collaborative authoring and centralized control/enforcement of standards. The collaborative authoring features are usually less important in this case, however, since these tools are generally simpler and easier to use, thus enabling non-technical staff to do the authoring without requiring teams of developers with specialized skills. Examples include:

- AuthorPoint®
  http://www.authorgen.com/
- Brainshark Rapid Authoring®
  http://www.brainshark.com/solutions/rapid-authoring

3.7.2. Desktop-based external document converter/optimizer tools

Examples of these applications include:

- Articulate Presenter®
  http://www.articulate.com/products/presenter.php
- Content Point®
  http://al.assima.net/contentpoint/index.html
- CourseAvenue Accessibility Player® [builds Section 508/ADA compliance into content]
  http://www.courseavenue.com
- Elicitus Content Publisher®
  http://www.elicitus.com/
- iSpring Suite®
  http://www.ispringsolutions.com/ispring-suite
- PowerPoint Integrator®
  http://www.solics.de/uploads/media/Lectora_2008_Lectora_Integrator_EN.pdf
- SmartBuilder®
  http://www.smartbuilder.com/product/smartbuilder
- Trivantis Snap!®
  http://trivantis.com/products/snap-e-learning-tool/
- Wimba Create®
  http://www.wimba.com/products/wimbacreate/
- Zenler Studio®
  http://www.zenler.com/

3.8. Intelligent Tutoring Systems (ITS)

ITSs are a new and rapidly emerging technology that, in their most robust implementations, use artificial intelligence to mimic the behavior of an expert human tutor, including holding a naturalistic (often inductive, Socratic question-based) dialogue with the student via a 3D avatar. Other ITSs provide step-based assistance when solving problems, some using a text-based interface. For more information, see the Wikipedia article at http://en.wikipedia.org/wiki/Intelligent_tutoring_system.

A key difference between ITSs and other forms of technology-based learning is that true ITSs dynamically generate instruction in real time through artificial intelligence algorithms. They do this in the form of dynamically crafted conversation (usually based on rule sets that adapt to the student's ongoing correct and incorrect expression of concepts) containing discussion, hints, feedback, questions, etc. The most robust systems can "understand" free text explanations of concepts given by the student based on questions from the system, and dynamically calibrate their pedagogical strategy and responses on an continuous evaluation of the student's understanding. This is fundamentally different from other technology-based learning, where the instruction is 100% predesigned, pre-developed, and pre-packaged, thus inherently limiting the interaction, response choices, and paths the student can take to learn the content.

ITSs are not completely devoid of this "prearranged instruction" approach, however, since: 1) questions to elicit knowledge and assess the student's understanding are usually planned in advance 2) some level of anticipation of student responses to questions (and consequent rule sets for dealing with them) currently is required to be programmed into ITSs; 3) the AI "understanding" module of the ITS needs to be "trained" in the content; and 4) ITS courses usually contain one or more pre-developed content modules, which could be standalone, separate eLearning courses, tutorials, or media objects such as text, graphics,

animations, simulations, videos, etc. These content modules are adaptively delivered by the ITS to convey initial information or concepts to the student, or reinforce or remediate understanding later in the instruction.

Each of the three items described above represent separate authoring dimensions; #1, 2, and 3 are usually an integral part of the ITS and are completely dependent on the capabilities and engineered design of the system. This authoring is done mostly as custom programming by system engineers.

In the case of dimension #4, the course author must design and develop these content modules in advance separately from the ITS (using any of the authoring tools described elsewhere in this document), and link them to instructional nodes programmed into the ITS (described in dimension #2).

There are currently no universal standards that would allow interoperating between an authoring tool and an ITS. However, there is conceptual movement towards this interoperable separation of authoring function from the ITS. Examples include the following:

- ASPIRE
  http://aspire.cosc.canterbury.ac.nz/ASPIRE.php
- Autotutor Lite
  http://www.skoonline.org/
- Cognitive Tutor Authoring Tools
  http://ctat.pact.cs.cmu.edu/
- The Extensible Problem-Specific Tutor (xPST) System [open source]
  http://xpst.vrac.iastate.edu/
  http://code.google.com/p/xpst/wiki/xPST_System
- Generalized Intelligent Framework for Tutoring – GIFT [open source]
  https://www.gifttutoring.org/projects/gift/wiki/Overview
- Mobile Coach® [works via text messages sent by the ITS]
  http://mobilecoach.com/
- Oppia [open source – not a full blown ITS, but has some elements of one]
  https://code.google.com/p/oppia/
- Rashi
  http://althea.cs.umass.edu/ckc/40rashi.html
  http://rashi.cs.umass.edu/
- SimCore®
  http://www.stottlerhenke.com/products/SimCore/simcore.htm
- Task Tutor Toolkit®
  http://www.stottlerhenke.com/products/ttt/task_tutor_toolkit_overview.pdf

3.9. Auxiliary tools

3.9.1. eLearning assemblers/packagers

These tools assemble objects authored in other tools into an organization/sequence of learning objects, usually to create SCORM packages (see 4.10.1. *SCORM* for more information). For those that produce SCORM packages, most provide the ability to:

- Package the content

- Author the manifest
- Validate conformance
- Provide a test run-time environment
- Insert and edit SCORM data model elements
- Enter metadata

Some allow programming of SCORM 2004 sequencing and navigation as well.

Examples include:

- eXact Learning Packager®
  http://www.exact-learning.com/en/products/learn-exact-suite/exact-packager-scorm-compliant-content-authoring
- Frameworker for SCORM®
  http://www.i-a-i.com/view.asp?aid=292
- RELOAD Editor [open source)
  http://edutechwiki.unige.ch/en/Reload_Editor
- SCORM Developer's Toolkit®
  http://www.e-learningconsulting.com/products/SCORM-source-code.html
- SCORM Driver®
  http://scorm.com/scorm-solved/scorm-driver/
- Trident®
  http://www.scormsoft.com/trident

3.9.2. Specific interaction object creation tools

These are generally stand-alone or accessory application modules, often sold either individually or in application suites; vendors design each module to produce a specific interaction. You generally purchase modules to meet specific interaction needs that are impossible or difficult to create in your primary tool; you use these tools to create the interactions, and then assemble/integrate the code objects into your eLearning course in the primary authoring tool (including PowerPoint presentations). Some have "linker" applications that allow linking them together without the need for a primary authoring tool container. Many of the interactions in these tools are geared towards assessment.

This SaaS (software as a service) model has been creeping into the authoring tool space at a slower pace than have LMSs, since the latter are such complex, large systems with higher potential savings from using the SaaS model. Examples include:

- Acuity Performance Task System®
  http://www.ctb.com/ctb.com/control/main
- eActivity®
  http://www.epathlearning.com/
- Exam Engine®
  https://www.plattecanyon.com/
- Hot Potatoes™ [freeware]
  http://web.uvic.ca/hrd/hotpot/

- iSpring Quizmaker®
  http://www.ispringsolutions.com/ispring-quizmaker
- Perception®
  http://questionmark.com
- Quiz Creator®
  http://www.sameshow.com
- Quiz Maker®
  http://www.proprofs.com
- QuizMaker®
  http://www.articulate.com
- Studio 09®
  http://www.articulate.com
- Raptivity®
  http://www.raptivity.com/
- ZEBRAZAPPS®
  http://www.alleninteractions.com/products/zebrazapps
- Wolfram Problem Generator™ [generates math practice problems]
  http://www.wolframalpha.com/pro/problem-generator.html

3.9.3. Media asset production and management tools

These tools create graphics, audio, video, and animation files. Note that there is a hybrid media category emerging called “flex media”. This approach bridges the gap between a still image and a video. It can provide some amount of motion (limited to a certain part of an image), helping to tell a story that would normally require full motion video, but avoiding the file size, production complexity, and cost of full motion video. Flex media often take the form of “cinemagraphs” (see http://en.wikipedia.org/wiki/Cinemagraph) and “photospheres” (see http://www.androidcentral.com/create-your-own-street-view-photo-spheres).

Examples of media asset production and management tools include:

- 3DS Max® [3D graphics/models]
  http://www.autodesk.com/products/autodesk-3ds-max/overview
- Adobe Presenter® [video capture)
  http://www.adobe.com/products/presenter.html
- Audacity [sound production – free, open source)
  http://audacity.sourceforge.net/
- Audition® [sound production)
  http://www.adobe.com/products/audition
- Adobe Voice® [voiceover animation)
  https://standout.adobe.com/voice/
- Bryce® [3D landscape modeling)
  http://www.daz3d.com/i.x/software/bryce/-/
- Camtasia Studio® [screen recording to create system training, demos, etc.)
  http://www.techsmith.com/camtasia/

- Cbox® [video capture and streaming platform]
  http://www.winnov.com
- Edge Animate® [HTML 5-based web animations]
  http://www.adobe.com
- Final Cut Pro® [video editing]
  http://www.apple.com/finalcutpro/
- Fireworks® [web graphic optimization]
  http://www.adobe.com/products/fireworks.html
- Flash® [animations and interactive objects]
  http://www.adobe.com/products/flash/
- Flipagram® [creating video "short stories"]
  http://flipagram.com/
- Garage Band® [music creation]
  http://www.apple.com/ilife/garageband/
- GoReact® [time-coded feedback and critique of speeches, presentations, or other performance-based skill]
  http://www.goreact.com
- Hapyak® [interactive video]
  http://www.hapyak.com
- Hyperlapse® [creating time-lapse videos]
  https://hyperlapse.instagram.com/
- Instagram® [micro-video creation]
  https://instagram.com
- Illustrator® [synthetic, line art graphic editing]
  http://www.adobe.com/products/illustrator/
- iMovie® [video editing]
  http://www.apple.com/ilife/imovie/
- Kaltura [cloud-based video production platform – open source)
  http://corp.kaltura.com/
- Logic Pro® [music production]
  http://www.apple.com/logicstudio/
- Multipop® [interactive video]
  http://www.getmultipop.com/
- Nawmal® [AI-driven video scenario creation]
  http://www.nawmal.com/
- Photoshop® [photographs or continuous tone graphic editing]
  http://www.adobe.com/products/photoshop/family/
- Poser® [human 3D model creation]
  http://poser.smithmicro.com/

- Rapt Media® [interactive video]
  http://www.raptmedia.com/
- Simpleshow® [a service that creates "explainer videos" that visualize content through animations that synch with the audio track]
  www.truscribe.com
- Snag It® [screen capture]
  http://www.techsmith.com/snagit-gslp.html?gclid=CNK6gceWyLcCFQyk4AodAV8AFw
- SWiSH Max® [animation – outputs to Flash format]
  http://www.swishzone.com/index.php?area=products&product=max
- Truscribe® [a service that creates "graphic recordings" that visualize content through drawings that appear and evolve in synch with the audio track]
  www.truscribe.com
- VideoScribe® [an automated tool that simulates the creation of "whiteboard videos"]
  http://www.videoscribe.co/
- Viddler® [platform for interactive video]
  www.viddler.com

3.9.4. Word processors, page layout, and document format tools

These tools create eLearning reference and planning documents and store them in a convenient format (for example, PDF) that preserves their appearance. Examples include:

- Acrobat®
  http://www.adobe.com/products/acrobat/
- Google Docs®
  http://docs.google.com
- iBooks Author®
  http://www.apple.com/ibooks-author/
- Mindjet Maps® [visual mapping tool for organizing content – for iPad)
  http://itunes.apple.com/us/app/mindjet-maps-for-ipad/id440272860?mt=8
- Mindmeister® [visual mapping tool for organizing ideas – for authoring, or direct use by students)
  http://www.mindmeister.com
- Mockflow® [user interface concept/wireframe design tool)
  http://www.mockflow.com
- Office®
  http://www.microsoft.com/
- OpenOffice [open source)
  http://www.openoffice.org/
- QuarkXPress®
  http://www.quark.com/
- Viewportsizes.com® [guide to screen sizes of mobile devices]
  http://viewportsizes.com

3.9.5. Database applications

These applications create and configure databases that may be accessed by an eLearning application. Examples include:

- Access®
  http://office.microsoft.com/en-us/access/FX100487571033.aspx
- Oracle®
  http://www.oracle.com/

3.9.6. Web-based collaboration tools

These applications create collaboration mechanisms and peer-to-peer communication functions, normally for meetings. Use of these features in eLearning is increasing rapidly; some vendors now specifically tailor these tools to support eLearning and their authoring and delivery systems. There is considerable overlap between these tools and the virtual classroom authoring tools category described in 3.3. *Virtual classroom systems*. Examples include:

- Adobe Connect®
  http://www.adobe.com/products/adobeconnect.html?promoid=DINSD
- ELGG [open source]
  http://elgg.org/
- GoToMeeting®
  http://www.gotomeeting.com/fec/online_meeting
- WebEx®
  http://www.webex.com

3.9.7. Web page editors

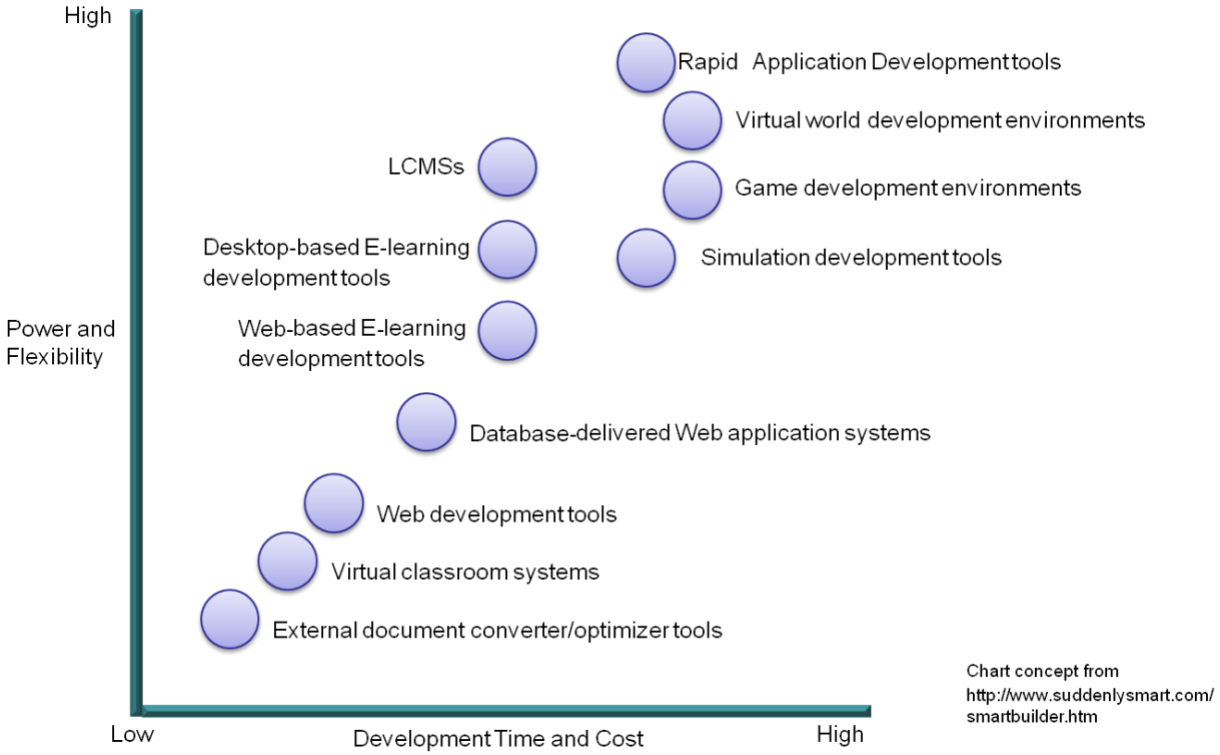
These applications create web pages that may be references or are otherwise ancillary to the main eLearning course screens. The tools listed in 3.1.1. *Website development tools* are also used to create web pages; the tools listed here differ in that they do not include extensive site management features, hence they are simpler to use and less expensive.

- CoffeeCup®
  http://www.coffeecup.com/html-editor/
- Easy WebContent®
  http://easywebcontent.com
- Editor®
  http://www.mozilla.org/editor/

3.10. Comparison of categories

Although eLearning projects can vary widely in their level of effort and complexity, thus putting varying burdens on the capabilities of authoring tools, the following chart illustrates one way in which some of the categories of tools presented in this section can be compared. Much of the variation between categories is simply due to the inherent complexity of the product that the category is designed to author. Note that in general, as power and flexibility increases, so does the development time and cost (both of the tool and the products it creates).

According to a recent survey of authoring tools (Shank & Ganci, 2013), users generally prefer power (58.8%) over ease of use (36.6%).



A 2014 survey by DiDonato (2014) compared new eLearning tools being acquired. These numbers represent respondents reporting that they were in the process of acquiring tools in the specified category. Note that some of these are not authoring tools per se (although there is an authoring component involved).

Mobile learning 33%
e-Learning development tools 30%
Video solutions 25%
Virtual events/classroom 22%
Assessment and testing 21%
Gamification and reward solutions 18%
Presenter tools 17%
Content development tools 17%
Wikis, blogs, or forums 16%
Collaborative work spaces 15%
Social networks 15%
Web conferencing 13%

4. Special features and issues to consider

4.1. Rapid eLearning authoring tools

Rapid e-learning authoring tools are simpler to use than the standalone tools described in 3.1.3. *eLearning development tools*, since they enable developers to use a familiar program like PowerPoint® or Word® for authoring the raw content. Developers then convert these documents to eLearning screens using the rapid eLearning authoring tool, adding interactive features. These tools enable the rank and file user (this means subject matter experts with no page design, authoring, or programming experience—not instructional designers or course developers) to create training modules.

The general advantages often cited for these tools are:

- Shortened development times
- Reduced cost, due not only to the reduced overall level of effort (LOE) in producing the course, but because the authoring tools are generally cheaper
- Ability to quickly and easily make changes and redeploy content (especially critical where content is volatile)

Critics of these tools note that they often produce courses with a cookie-cutter look and feel (due to the template-driven architecture and technical inexperience of course developers) and that there is usually a substantial tradeoff in the ability to produce complex eLearning products, as shown in 3.10. *Comparison of categories*. The fact that developers must use Word® or PowerPoint® as a starting point can be a limitation for producing interactive, media-rich eLearning, unless the tool provides a substantial ability to add interactions after conversion. In these cases, the fact that the output format is a robust interactive file type like Flash, can be helpful in enabling greater interactivity in the final product.

These tools generally make the most sense where content already exists in a usable form (for example, as PowerPoint® slides used in an instructor-led course) and where only lower-level learning objectives need to be met. Note that most PowerPoint® slides contain highly abridged information that is not designed to stand alone as learning, so there will need to be some effort to modify the slides or augment them before or after conversion.

A lively debate on the pros and cons of use of these tools is presented in the first three chapters of Allen (2012). Also note that the definition of the term "rapid eLearning authoring tool" is defined differently than the way it is described above; some define it as any tool that does not require the author to write code or script to create content. However, ADL argues that many of those tools, despite their careful attention to ease of use and WYSIWYG design, have become so complex that they have lost their claim to the "rapid" moniker. ADL thus defines tools in this category as ones that are predicated on using PowerPoint or Word as a starting point, even essentially optimizing PowerPoint files for eLearning without changing them to another file format.

Use of these tools has become quite widespread; the eLearning Guild reported in their 2013 survey of authoring tools (Shank & Ganci), 2013) that 64% of respondents create "PowerPoint-to-eLearning" with their authoring tool.

4.2. mLearning authoring tools

Authoring learning for mobile devices is different in a number of significant ways, involving the following:

- Operating systems and hardware specs (especially screen size and resolution) for mobile devices are very different from one device to the next, leading to the emergence of responsive design For more details, see 4.12 *Responsive design*.
- Connection speed to data networks is highly variable, depending on time of day, user location, etc.
- Performance is generally considerably less powerful than desktop computers. This is dependent on such things as memory, disk space, chip design, etc.
- Mobile phones are highly personalized (as opposed to desktop computers), which makes it hard to baseline a design.
- There are different paradigms for interaction with mobile device (e.g., using fingers, especially thumbs, rather than a mouse). This presents problems for rollover interactions and large text entry windows.
- Many phones can dynamically shift portrait vs landscape orientation. Content may need to adjust accordingly, or be viewed only in a locked mode (which users need to be warned about).
- There is a need to test developed content on many different platforms—small businesses do not normally have the resources to acquire all of these. A way to avoid this problem is to use http://www.deviceanywhere.com for testing (a cloud-based mobile phone emulator that shows you what your learning looks like and how well it works on any platform). Note that emulators are not always 100% consistent with the actual device.
- Not all eLearning content is appropriate for mobile delivery. Appropriate mobile learning content can be thought of as falling in three basic categories (Quinn, 2011):
    - Learning augmentation
        - Motivational examples
        - Extending learning processes with new concept representations, new contexts for examples, extended practice.
    - Performance augmentation (aka performance support)
        - Decision support tools
        - Job aids
        - Help applications
    - Learnlet/Microcourses
- Output files can be standalone apps or browser-delivered. In the case of standalone apps, Flash® is currently not compatible with some devices.
- Authoring tools currently do not usually have the ability to build content to take advantage of built-in functions like cameras, compasses, GPS, accelerometers, gyroscope, and other sensors, although this is changing quickly. Certain mobile device functions will always be hard to access unless you build your content within a native app environment (using a low level programming language like Objective C, however).
- Standard interactive controls on the mobile platform work differently than on the desktop. This has resulted in the following best practices:
    - Don’t use radio buttons, use regular buttons
    - Don’t use rollovers
    - Use built-in cell phone themes, functions, and navigation when possible

An important decision in authoring mLearning is whether to author the application as a standalone app or a web-based application that displays with the mobile device's browser. Here are some of the considerations (Haag, 2011):

- Develop mLearning as a web application when:
  - You seek cross-platform compatibility.
  - You can't support the development of native apps using proprietary Software Development Kits (SDKs).
  - Accessibility is a requirement.
  - Using more advanced capabilities of the device isn't required (e.g., offline, camera, accelerometer, gyroscope, etc.).
- Develop mLearning as a native app when:
  - You are charging for it (for profit).
  - You are creating a game.
  - You are using specific location information.
  - You are using cameras, accelerometers, etc..
  - You are accessing the file systems.
  - There will be offline users.

The following is a list of pros and cons for each:

- Native apps
  - Pros
    - Best for:
      - Developing games and 3D
      - Using cameras or other features (e.g. Augmented Reality)
      - Accessing the file systems
    - Do not require connectivity, though can be designed to connect to any kind of cloud service or web application.
    - Faster, more predictable performance because they can directly access the mobile device's hardware.
    - They offer a best-in-class user experience, offering a rich design and tapping into device features.
    - Can be much more secure
    - Can lock or delete content remotely
    - Relatively simple for a programmer to develop for a single platform
    - Content can be pushed or pulled.
    - Content can be downloaded for offline consumption
  - Cons
    - They require a unique programming language.
    - They cannot be easily ported to other mobile platforms.
    - Updates generally require downloading a new file.
    - Accessibility is difficult to achieve.

        - Can take longer to deploy
        - Developing, testing, and supporting multiple device platforms can be costly.
        - They may require certification and distribution from a third party that you have no control over (esp. in the case of Apple).
- Web apps
    - Pros
        - Languages are easier to learn and use: HTML, CSS, and JavaScript
        - Simple to deploy across multiple handsets – don't need separate versions for each operating system
        - Easier to control access (since not available through app stores)
        - Features are consistent with features of a PC browser (thus user experience is similar)
        - Content is accessible on any mobile web browser
        - Can be packaged & ported as a hybrid app using a tool like Phone Gap (see below)
        - Allows potential access to the same LMS environment (or a specialized flavor of it) that desktop offers
        - Faster to deploy and update
    - Cons
        - Requires live Internet access
        - Optimal experience might not be available on all handsets.
        - Browser support for interactivity and rich media varies. This forces providers to go to least common denominator.
        - Can be challenging (but not impossible) to support across multiple devices
        - Don't always support native application features, like offline mode, location lookup, file system access, camera, etc.
        - Can't push content to the learner. Must notify them via some other communications media that there is new content that they can pull.

If you are targeting more than one device, it is important to consider a cross-platform, hybrid approach, allowing you build once rather than coding separately for each device. To do this, you develop a mobile web application first, enjoying all of the benefits of that approach, as described above. Then, you can gain most of the benefits of a native app by "wrapping" the web app in an application shell so that it is treated by the mobile operating system as a native app. It can then be distributed through the app stores and downloaded on your device like any other native app. One of the downsides to this is that the cons listed above for web apps still remain (except for requiring connectivity, since the app is not served from the network but stored on the device). It is important to keep in mind that performance of such hybrid apps is generally less compared to a pure native app approach.

Application shells for hybrid apps include:

- Rhodes (http://rhomobile.com)
- PhoneGap (http://phonegap.com)
- Titanium (http://appcelerator.com)

- Adobe AIR (http://adobe.com)
- MoSync (http://www.mosync.com)

It is important to understand that these are development environments, not authoring tools (although now some are integrated with authoring tools, such as PhoneGap with Captivate); they require programmer talent to be used effectively. For a detailed treatment of the features and pros and cons of each of these, see Udell (2012), pp. 202 – 210.

There is a growing consensus in the mLearning community that a course that is being developed for both mLearning and desktop delivery should be developed for the mobile platform first, then modified for the desktop version, since this tends to drive simplifying the content. Authoring the mLearning version first is called "progressive enhancement," as opposed to authoring the desktop version first, which is termed "graceful degradation."

Rather than automatically including every screen in both platforms (desktop and mobile), some authoring tools (such as the EXPERT Platform) allow the author to designate only certain screens for mLearning delivery. This acknowledges the fact that mLearning tends to work better for short, concise content objects. These tools can also use a more performance support-oriented template for the mLearning version as well.

Authoring tools are starting to appear that offer alternative formats that are dynamically determined by the content when it comes in contact with the mobile device. For example, Articulate Storyline® detects whether the user's device can display Flash files, and will deliver content in that format if so. If not, it will deliver it in HTML 5. And if the user has an iPad® that has the Articulate app, it will deliver content through that player.

As a hybrid approach, you may want to consider placing QR codes in your eLearning or printed content materials, that mobile devices can read to provide a convenient entry point to a specific piece of reference information. QR codes are matrix barcodes that link to a web URL.

See the following ADL documents for more information on mobile learning:

- mLearning Guide (optimized for access from a mobile phone)
  http://mlearn.adlnet.gov/
- mLearning Handbook (for desktop computer use—contains more detailed information)
  https://sites.google.com/a/adlnet.gov/mobile-learning-guide/home/

4.3. Open source, freeware, and GOTS solutions

Open source software is defined by the fact that its source code is made available for no cost, and users are licensed to study, change and distribute the software to anyone and for any purpose. It is usually developed in a public, collaborative manner, and there is usually a community site where contributors can share and discuss their contributions.

Open source options are obviously attractive to buyers because there is no licensing cost involved. You need to be clear on the pros and cons of purchasing an open source solution, as in the long run, the cost could equal or exceed a commercial solution. It's easy to be over-enamored with the free license aspect and ignore other aspects of the solution that will cost money regardless. Those aspects generally include installation, customization, and support.
It is also easy to overlook the potential advantage of open source tools in that the product can be completely tailored to the particular requirements of the organization. If managed properly, this advantage can make an open source solution cheaper, not just because the license is free, but because the development and customization efforts can be focused solely on the needs of the organization and nothing more. Contrast this with a commercial product with lots of features that your organization may not need

(but which you are essentially paying for nonetheless). The business model for a standard commercial system is to build to the widest set of possible requirements to attract the widest client base. Your organization may not need all or even most of these requirements.

On October 16, 2009, U.S. DoD issued new guidance on open source software (see http://powdermonkey.blogs.com/files/2009oss.pdf). The guidance emphasizes that open source software should have equal weight as proprietary software during acquisition evaluations. It is a break from the past, when open source software was deprecated for use in DoD due to security and quality concerns. The benefits of open source software are described in this guidance document as follows (open source software is referred to as "OSS"):

- The continuous and broad peer-review enabled by publicly available source code supports software reliability and security efforts through the identification and elimination of defects that might otherwise go unrecognized by a more limited core development team.
- The unrestricted ability to modify software source code enables the Department to respond more rapidly to changing situations, missions, and future threats.
- Reliance on a particular software developer or vendor due to proprietary restrictions may be reduced by the use of OSS, which can be operated and maintained by multiple vendors, thus reducing barriers to entry and exit.
- Since OSS typically does not have a per-seat licensing cost, it can provide a cost advantage in situations where many copies of the software may be required, and can mitigate risk of cost growth due to licensing in situations where the total number of users may not be known in advance.
- Open source licenses do not restrict who can use the software or the fields of endeavor in which the software can be used. Therefore, OSS provides a net-centric licensing model that enables rapid provisioning of both known and unanticipated users.
- Since OSS typically does not have a per-seat licensing cost, it can provide a cost advantage in situations where many copies of the software may be required, and can mitigate risk of cost growth due to licensing in situations where the total number of users may not be known in advance.
- By sharing the responsibility for maintenance of OSS with other users, the Department can benefit by reducing the total cost of ownership for software particularly compared with software for which the Department has sole responsibility for maintenance (e.g., GOTS).
- OSS is particularly suitable for rapid prototyping and experimentation, where the ability to "test drive" the software with minimal costs and administrative delays can be important.

  (Memorandum Clarifying Guidance Regarding Open Source Software (OSS), Oct. 16, 2009)

What is important to understand about open source software is the relationship it behooves you to build with the community surrounding the open source product you are acquiring. Staying in touch with the community in order to be able to discover and use already-developed modules of functionality that you need (which are not part of the product baseline) can decrease your customization costs enormously. Open source communities often remind you that deploying open source means you are a responsible member of their community. There is an expectation that you must contribute to—as well as receive from— the community code, training, and documentation. The cost of staying active in the community and both researching and acquiring as well as sharing your products and solutions must be factored into the LOE for acquiring an open source tool.

It is also important to evaluate the strength and size of the open source community for the open source product you are acquiring, as well as the longevity of the product. This can mitigate obvious concerns that

major sponsors of open source software can stop development at any time, or that communities can atrophy. Another possible concern is that a tool can grow so quickly in its popularity that documentation takes a back seat to development and has not caught up to the current release of the software. Especially in the case of open source software, where you have no vendor who is obligated to support you, a lack of adequate documentation can make a product difficult to install, use, maintain, and troubleshoot.

Finally, the baseline versions of some open source products are very basic; some level of customization is often needed to make the software not only meet your special requirements but also meet a modest level of universally recognized functionality for the type of product. It may be risky to assume that an open source product will be usable straight out of the box. If you have no development resources ready to augment the product's functionality right after you acquire it, you may not be able to use it for some time.

Freeware may or may not also be open source. Freeware may have restrictions on copying, distributing, and making derivative works of the software, where open source software does not. And freeware does not necessarily make source code available. Freeware may be restricted to personal use, non-profit use, non-commercial use, etc. Freeware that is not open source is a risky investment, since you cannot easily customize it.

There may be special restrictions on use of freeware within your organization. For U.S. DoD, see http://www.acq.osd.mil/ie/bei/pm/ref-library/dodd/d85001p.pdf

GOTS only applies to government entities. GOTS software can be created either by the technical staff of a government agency or by a commercial vendor (usually the latter). GOTS systems usually have the following characteristics:

- The government has direct control over most aspects of the product, including the source code.
- The vendor or creator has given a license to the government entity who paid for it to freely use and share it within the government. The license does not permit the government to give or sell it to outside entities.

A popular model for GOTS installations is to have regular meetings where representatives from organizations that use the product throughout government discuss new requirements and possible new features. At these meetings, agreements are made between the representatives about sharing the cost for adding these features (which, after they are developed, are available to all users).

The original vendor/developer is usually the preferred entity for doing the customizations, since their developers were directly involved in creating it and have the most knowledge about working with the code base. This pre-existing experience and expertise can substantially reduce the cost of further development and customization. A GOTS license does not stipulate that the original vendor has to do the customization, however.

4.4. Hosted solutions

Some vendors of web-based authoring tools offer a hosted option. A hosted tool is installed and managed on the vendor's server by their staff, rather than behind your enterprise firewall by your staff. Some of the advantages of a hosted platform are:

- Eliminates the cost of hardware and network infrastructure needed to support a local installation of the system.
- Lowers staff costs for administration and maintenance.
- Puts less bandwidth load on the corporate network.
- Content and feature updates can be accomplished without intervention by staff.
- Enables faster implementation.

- Requires little or no internal technical support or development.

One of the main disadvantages of a hosted solution is that it restricts opportunities and scope for local customization. Also, a hosted solution may not provide the level of security required by your organization, although hosted solutions are increasingly more secure. Finally, it may not be an option for government entities, since government rules tend to mandate outright ownership and control of systems, while a hosted solution resembles leasing.

Vendors who offer hosted solutions commit themselves to a robust hosting and networking infrastructure with uninterrupted, 24/7 access from any location. The system that they host must be scalable and have redundant backup and security. These are items for due diligence verification during the acquisition process.

Most hosted solution vendors offer access to their tools on a subscription basis. And some of these, such as easygenerator®, offer free accounts to produce a limited number of courses, as a gradient towards fee-based accounts.

4.5. Templates, themes, and skins

One of the most dramatic things you can do to streamline your workflow and reduce level of effort (LOE) is to use templates, themes, and skins. You may hear the terms “skins,” “themes,” and “templates” used interchangeably, but it is useful to think of them as differing in the following ways.

Skins usually comprise an interface wrapper or design that is applied dynamically to a basic content layout. The LMS usually provides the skin from a library of them and controls the skinning process. Skins are very common in cases where different organizations or user groups are sharing the same LMS, and they need to use different “storefronts” to reinforce/brand their identity, or at least in order to not confuse users as to the owner or source of the course. The LMS detects what organization, role, etc. the user belongs to, and dynamically skins the content. Skins enable local variations on parent content objects, providing each organization or learner community with its own visual interface or style for the same base content, managed on the level of a single master copy.

Themes are generally style sheets that globally control the appearance and format of screens, but are static and configured in the content during the process of authoring, via the authoring tool. Unlike skins, themes provide formatting for items appearing within the interface (often in addition to the interface itself). They are not applied dynamically by the LMS. Themes generally control elements at a more fine-grained level, controlling color themes, look and feel of interactive widgets, etc.

Templates are of two types. One supplies a convenient starting point for developing a screen (often including the interface); you simply replace placeholder text, graphics, etc. Templates can include not just visual elements but a large proportion of the functionality of the screen, often based on instructional activities or kinds of interactions. Some authoring tools force you to use templates as starting points for building screens; you cannot design an individual page without specifying a template first.

Templates can also be a whole structure for a course. For example, you may have a pre-test with a certain number of questions, followed by instruction, followed by a post test. In this kind of template, the standardized sequence of activities of the course rather than the appearance is controlled. This kind of template can also be provided by the LMS, though in this case, the elements have to be created in the authoring tool as separate learning objects.

An authoring paradigm that relies heavily on templates, sometimes termed “form-based authoring,” is popular for rapid eLearning development (see 4.1. *Rapid eLearning authoring tools*). Under this paradigm, the course author populates forms with content data and objects. There is a form (i.e., template) for each type of screen, built to accomplish a specific design, function, and/or interaction. The form is limited to the functions and designs included in that template. This is contrasted with “freeform

authoring," where authors start with a blank screen, and have unlimited access to all of the functionality provided by the tool itself. Form-based authoring takes little or no programming skill and enables authoring by non-technical SMEs, while freeform authoring takes some programming skills. In form-based authoring, instructional designers or SMEs simply choose the template or theme that applies to a screen they wish to build and populate the content. This saves huge amounts of time, reduces the requirement for technical expertise, and simplifies the authoring process, since authors are just populating a template rather than engineering the whole screen.

During the design phase of a project, authors may either develop their own themes and templates or choose them from a library (usually packaged with the authoring tool). Skins are usually controlled and customized via a function within the LMS. Managers can mandate the use of templates and themes to enforce uniform standards for eLearning across an organization.

Tools vary in the features they provide for building your own templates. In general, the process is:

1. Create a skeleton for the template that specifies the types of media objects that will be included on the page.
2. Create or specify the user interface within which the content will appear (often in terms of a pre-built skin).
3. Create the layout for the template that includes sizes and positions of media objects.

Of course, use of templates can restrict creativity and create eLearning that is "cookie cutter" in look and feel; developers can mitigate this by simply populating a template/skins library with a wide selection of appropriate designs and screen types.

Screen templates are critical for authoring mobile learning, since screen real estate is so restricted and particular to each platform. Some authoring and graphic design tools include a catalog of templates for particular devices.

4.6. Security considerations

This section only applies to web-based tools. Like any other enterprise system, authoring tools must meet the security needs of the organization. For commercial installations, authoring tool security amounts to:

- Protecting against unauthorized login. This is not primarily a function of the tool, whose login functionality relies on universal web standards, but rather the placement of the system within the corporate intranet environment and the inherent security features of that placement. Commercial entities are of course concerned about other organizations gaining competitive advantage by seeing the training of competing companies, and government entities have obvious security concerns, so access to the tool is a primary concern.
- Locking users out of capabilities that are not included in their user profile, in other words, keeping users from doing particular things once in the system that they are not authorized to do. All web-based authoring tools include levels of permission based on roles, but beyond this, they vary widely in terms of the types and number of roles and permissions that can be assigned.
- Segmenting system permissions so that they map to the levels and specific kinds of permission that your organization requires. The question here is, if the system forces you to use a permission/roles assignment template, how applicable is it to your environment, and can templates be tailored to meet your needs? Is there an override that permits assignment of individual permissions on a function-by-function basis?

For DoD organizations, there are specific considerations relating to the possible harmful effects to national security and individuals' life and limb due to unauthorized access to the system and particular courses that may be classified, etc.. There are a number of issues that need to be considered in this regard.

4.7. File formats

4.7.1. Input

Sometimes there is a need to convert materials from ILT to eLearning. The input format in this case is often Word® and PowerPoint® files. Rather than starting from scratch to recreate these in the output medium, you can use an external document converter/optimizer tool (described in 3.7. *External document converter/optimizer tools*) to automate the conversion into eLearning. The first step is to convert the PowerPoint® and Word® documents to web pages residing within an eLearning interface; developers can then build eLearning interactivity into these pages (some interactive features may be automatically converted from PowerPoint® as well).

If you are in this situation, you need to: 1) limit your tool selection to this specific kind of tool; and 2) carefully assess your input formats to confirm that they match the tool's support input formats. Do not assume that your legacy ILT documents are in Word® and PowerPoint®; some may be in desktop publishing applications like Quark®, or may only be available in PDF format (and the original source files may be lost).

4.7.2. Output

Probably the single most important question to ask when choosing an authoring tool is: "What output file format(s) does it produce"? It is important that you determine your output format before beginning to choose authoring tools. This serves to filter and focus your search considerably, and ensures that: 1) the files will work within your IT and training delivery infrastructure, including the end-user platforms (for example, PC and Mac), operating systems, and browsers; and 2) you are not stuck with a proprietary format that may disappear from the marketplace and eventually leave you with no ability to open and edit the files, nor with the ability to run them in a browser. Countless organizations that used Authorware® as their authoring tool now find themselves in this situation.

Many LCMSs and some web-based authoring tools are designed to assemble and deliver eLearning dynamically, at runtime. They can output complete eLearning files for use on another platform as a convenience, but are generally not intended to be used this way. However, one major LCMS vendor has reported to the authors that most of their clients prefer to generate files in advance and use them on a another platform (often, the same vendor's separate LMS product) rather than take advantage of the internal ability of an LCMS to dynamically assemble and deliver files at runtime. Even though runtime output files may not be stored on the server, it is important to consider the file format of the files that are delivered to users at runtime relative to factors such as browser compatibility.

Frequently, the type of training you are creating drives the output format. The most important division is between synchronous vs asynchronous learning. Authoring systems differ markedly in their optimization for these types of learning. For instance, for synchronous eLearning, an external document converter/optimizer tool is probably your best choice. Most of these tools offer outputs for synchronous learning such as speaker notes, handouts, or student guides, in Word® or PowerPoint® format.

The choice of an output format depends primarily on the requirements of your delivery system, as well as on the type of learning that the file format supports. For instance, the Flash® .swf format robustly supports high levels of interactivity and is supported by most delivery systems when embedded in web pages. However, the range of tools that can natively edit .fla source Flash® files is much narrower than DHTML.

You will need to check with your IT department to verify the compatibility of desired output formats with the network and firewalls. For instance, some firewalls block Java applets due to potential security risks. Output formats that require browser plug-ins other than ones that are provided as a default with the browser installation (such as Flash® with Internet Explorer®) can be a serious liability. This can put an administrative burden on users and/or IT personnel to install and maintain these plug-ins. They may be prohibited in a user environment for this or security reasons.

Some output formats (such as Flash® and HTML5) have the advantage of built-in compression and streaming of files at run time. This can be a significant advantage in cases where bandwidth is limited.

Output file format can greatly affect the editability of your developed course within other authoring tools. This can become an issue when a tool disappears from the marketplace or if a new author comes on board who prefers working in a different tool; if the course can be imported into another tool and manipulated as source files in that tool, these problems can be alleviated.

It is important to understand that authoring tools often use proprietary code objects (for instance, references to internal Java applets, or code inserted into HTML comment fields) to facilitate authoring functions and course features. There may be no problem running these courses in LMSs, and they may work in any browser, but these code objects may be difficult to understand, troubleshoot, and edit in another authoring tool or HTML editor. The ideal for an authoring tool is that the output format is identical to the internal source file format, and that this format is a clean, universal code like HTML; no proprietary code should be involved. This ensures that your code is "future proof", and will work with new versions of operating systems and browers.

It is important that you determine whether the files that an authoring tool produces (in a standard non-proprietary format) are 100% editable in other tools that say that they can handle that format, especially those that generate that format natively. For example, some authoring tools that allow output as source .fla Flash® files are not as fully editable as native Flash® files, although they could be opened in Flash®.

You may need to take a detailed look at a product's support of an output format. The term "supports" may not have the same connotations as the term "is optimized for" or "is built for".

Output formats are becoming even more important as we enter the mobile learning era. Authoring tools have features that support producing files that can play on mobile devices. In the past, developers had to completely customize eLearning architecture and format for mobile devices, but that is less often the case with devices like the Apple iPhone® that have robust browser capabilities (that match desktop browsers) and larger screens.

Note: at the time of this writing, the Flash format is not accepted on iPhones. Any authoring for the iPhone platform must take this into account. If you are developing courses for the desktop platform that you intend to repurpose for the mobile platform, you will have to re-output or rebuild any Flash pieces in your content in some other format (such as HTML5 or Adobe AIR®) for iPhone delivery.

As of this writing, HTML 5 is being implemented in the major browsers. This output format is likely to become a universal format for eLearning. See 7.18. *HTML5 format* for important considerations regarding this format.

4.8. Reuse of learning objects

Courses and the learning objects of which they are comprised are usually expensive to produce, no matter what authoring tool is used. This is especially true of media-rich learning objects. Of course, there is a natural incentive to reuse learning objects and media assets where it is instructionally appropriate, to save development time and money. Most authoring tools (especially web-based tools and LCMSs) acknowledge this fact by offering robust content object library functions that facilitate reuse within and

between courses. Reuse scenarios can range all the way from a single author with an authoring tool that offers a self-contained library of objects used (or on deck to use) in a course to a web-based tool that allows storage, search, and retrieval of objects across an entire enterprise of authors and courses. Reuse can even take place between organizations, enabled by registry efforts such as the Learning Registry (http://learningregistry.org/) and repositories such as the RUSSEL system (http://adlnet.gov/russel/).

Be aware that having an authoring tool that is optimized for reuse is only the first step towards realizing reuse as an authoring paradigm; institutional and logistical barriers may deter full implementation of reuse. Some examples of these barriers listed on the ADL RUSSEL site at http://adlnet.gov/russel/are as follows:

- Developed content is not stored in approved, accessible content repositories
- Creating, uploading, and maintaining metadata associated with learning assets, Shareable Content Objects (SCOs), or SCORM content packages is a time-consuming process
- Creating and registering content in approved content repositories (if available) is time-consuming
- How to optimize the design and development of eLearning content for reuse is not understood
- Policy enforcing reuse is either non-existent or ambiguous
- Intellectual property rights concerns

4.9. Commercially available courses

One option that may be less expensive than developing custom courseware from scratch is to purchase commercially available courses. Models for delivery vary; in some cases, they reside on the vendor's server only and require login to a separate LMS. A review of commercial courseware vendors can be found at http://mason.gmu.edu/~ndabbagh/wblg/Forbes-Review-Dabbagh.htm.

4.10. Standards support

4.10.1. SCORM

4.10.1.1. Overview

ADL has identified the following high-level attributes for all distributed learning environments.

- **Interoperability**: the ability to take instructional components developed in one system and use them in another system.
- **Accessibility**: the ability to locate and access instructional components from multiple locations and deliver them to other locations.
- **Reusability**: the ability to use instructional components in multiple applications, courses and contexts.
- **Durability**: the ability to withstand technology changes over time without costly redesign, reconfiguration, or recoding.

To achieve these attributes in distributed learning environments, ADL promotes the use of the Sharable Content Object Reference Model (SCORM). SCORM defines the interrelationship of course components, data models, and protocols so that learning content "objects" are sharable across systems that conform with the same model. To support interoperability, SCORM standardizes the means of communication from the sharable content objects (SCOs) to the learning management system (LMS), through an Application Programming Interface (API) and prescribed data model elements.

For more information on SCORM, see www.ADLNet.gov.

It is important to understand that SCORM neither dictates nor precludes any instructional, performance support, or evaluation strategy. SCORM does enable object-based approaches to the development and presentation of eLearning. This is enabled by aggregating learning content composed from relatively small, reusable content objects to form meaningful units of instruction. Individual content objects can thus be designed for reuse in multiple contexts, and aggregated variously to assemble new components and programs of instruction.

This object-based approach, intended to support reuse, means that content objects must not determine by themselves how to sequence/navigate aggregations that represent parcels of instruction. Doing so would require content objects to contain information about other content objects, which would inhibit their reusability. ADL addressed this requirement by standardizing a set of behaviors that that all SCORM 2004-conformant LMSs must support. Thus, the LMS, rather than the content, controls the movement of learners from SCO to SCO.

To support reuse, SCORM uses metadata to enable content objects to be discoverable through and across enterprises, within distributed content repositories.

NOTE: Content acquired by U.S. DoD must be SCORM-conformant ("current version") according to DoD Instruction 1322.26 (June 16, 2006). See http://www.dtic.mil/whs/directives/corres/pdf/132226p.pdf for more details.

4.10.1.2. Requirements for SCORM support

For an authoring tool to robustly support SCORM, the authoring tool must:

- Support object-based learning design
- Allow defining of SCOs at any level of organization
- Support incorporation of all SCORM data model elements into SCOs, including:
    - Mandatory calls inserted without consulting the developer
    - Optional calls inserted as drag and drop elements, such as a "Finish and Exit" button that triggers an *api.setValue(cmi.exit.normal); Terminate()* call.
- Create SCORM course packages that include all necessary files and information for the LMS to properly deliver the course at runtime. Either drop down menus or wizards should be available to assist the author in the process of creating the course package.
- Allow direct viewing and editing of manifest files
- Provide tools enabling reuse of course packages and manifests in creating new course packages and manifests
- Include a SCORM metadata editor. Ideally, some of this metadata is inferred or extracted from existing properties of the courseware, without requiring manual entry.
- Allow definition of sequencing and navigation rules for the course organization

Authoring tools support the SCORM requirements described above to widely varying degrees, with widely varying implementations. As part of your decision process, it is important to evaluate how fully the tools support each of these, and in what way. The depth of support for the standard can make a big difference in the LOE to produce conformant eLearning.

4.10.1.3. Recommendations to ensure SCORM conformance and support

Before you evaluate the authoring tools in terms of SCORM conformance and support, you should determine the target SCORM conformance level (for example, SCORM 2004 4th Edition) for your content. This will depend on the conformance level your LMS supports. LMSs can lag several versions behind the current level, and since SCORM levels are not all backward compatible (especially between SCORM 2004 and SCORM 1.2), it is important to determine the level of conformance of your LMS (and whether it is certified at that level).

SCORM comes in five versions:

- SCORM 1.1
- SCORM 1.2
- SCORM 2004 2nd Edition
- SCORM 2004 3rd Edition
- SCORM 2004 4th Edition (the current version)

If you expect to delivery legacy content along with content conformant with more recent SCORM editions, the tools need to include options for exporting these two standards separately.

Note that only content and LMSs can be defined as SCORM-conformant, and only LMSs can be certified as conformant. The operation of an authoring tool is not governed by SCORM, and many possible approaches to automating SCORM support exist. Only the content produced by it can be assessed for conformance, and even then, it depends on the configuration and parameters the author sets for the tool's output. Content may be conformant, but perhaps only if certain parameters are set in a particular way. This variability injects too much uncertainty in any determination of conformance. Therefore, authoring tools, unlike LMSs, are not judged to be conformant or non-conformant.

During the acquisition process, you will need to talk to your vendor or read documentation carefully to determine what the limitations are for creating SCORM-conformant content. For instance, some tools advertise SCORM conformance, but do not allow you to define SCOs at any level of course structure; you can only define the entire course as a single SCO. This defeats the ADL goal of learning object reusability.

We highly recommend that you acquire a sample SCORM-conformant eLearning course produced by the tool you are evaluating, and test it on your target course delivery system. Course delivery systems implement the same SCORM conformance level differently in some cases; the interaction of the particular implementation of SCORM in the course delivery system and the particular implementation of SCORM in your SCORM course package, even if both are at the same level of conformance, may uncover issues. This may impact your decision to purchase a particular tool.

If you do not have a target LMS available, you can use the ADL Sample Run Time Environment to see how your authoring tool's content output runs in a fully SCORM-conformant LMS (download from http://adlnet.gov/wp-content/uploads/2011/07/SCORM.2004.4ED.SRTE_.v1.1.1.zip).

You should also run a sample SCO produced by the authoring tool through the SCORM Conformance Test Suite (download from http://adlnet.gov/wp-content/uploads/2011/07/SCORM.2004.4ED.TS_.v1.1.1.zip)

to verify the level of conformance the authoring tool achieves in content output. As of this writing, the current version of SCORM is 2004 4th Edition.

One possible approach to comparing authoring tools for SCORM support involves creating the same SCO in multiple authoring tools and testing each one in turn in the SCORM Conformance Test Suite. Creating the same piece of content in various authoring tools compares the important task automation features of each tool—a comparison that necessarily involves subjective judgments—while "leveling the playing field" as much as possible. The test logs themselves augment the subjective aspects of testing by providing standardized and objectively developed records that depict each SCO's degree of success in supporting SCORM.

There is a SCORM Adopter listing (that includes authoring tools) on the ADL public web site. To see a list of authoring tools, go to http://adlnet.gov/wp-content/uploads/2014/09/SCORMAdoptersLocked.xlsx, enter **authoring tool** in the Keyword field, select other options as needed to see a filtered list, then click **Search**. These tools, along with many of the those listed in 3. *Categories and examples of authoring tools*, have built-in features to support achieving SCORM conformance; in most cases, however, some manual coding is necessary to create fully SCORM-conformant eLearning (for example, in the HTML/JavaScript "wrapper" for SCOs). Furthermore, many of these tools do not automate the creation of SCORM course packages (.zip files containing XML manifest files that describe SCOs, metadata, etc.). For this capability, you should use one of the tools listed in 3.9.1. *eLearning assemblers/packagers.* These tools automate the creation of SCORM packages, providing a GUI interface for configuring SCORM packages.

4.10.1.4. Support for SCORM 2004 sequencing and navigation

Currently, support for SCORM 2004 sequencing and navigation is rare among self-contained authoring systems. You may need to accomplish SCORM 2004 sequencing either by directly coding the manifest file (using XML) or by using a tool such as the RELOAD Editor (see 3.9.1. *eLearning assemblers/packagers*), which provides a GUI interface for defining sequencing and navigation. However, you should note that the terms and techniques needed to use the RELOAD™ Editor require a thorough understanding of sequencing and navigation concepts and logic under SCORM 2004. It is not for the "technically challenged."

Although most authoring tools do not provide the ability to generate SCORM 2004 sequencing rules from scratch, some allow you to choose from pre-built sequencing templates (the Reload Editor™ provides this, as well as the ability to define custom rules using a form with popdown menus allowing selection of rule operators).

4.10.2. Section 508

Section 508 (29 U.S.C. 794d) is a law enacted in 1998 that applies to all Federal agencies when they develop, procure, maintain, or use electronic and information technology. Agencies must give disabled employees and members of the public access to information that is comparable to the access available to others. For eLearning, this means, for example, that special features must be implemented in the content so that they are understandable using assistive software for blind persons. Achieving accessibility for deaf persons usually means including a script or closed captioning of any sound portions.

If your organization is subject to Section 508 compliance for eLearning products, it is critical that you include this as a decision parameter in your choice of an authoring tool. You can reduce the LOE in developing eLearning if your authoring tool(s) has built-in 508 compliance support, so that there is no need to undertake additional production steps (especially highly technical ones) outside the normal course of the authoring process within the tool. Your authoring tool should not only add elements to produce compliant eLearning code as you work with it, it should also prevent you from doing things that would produce non-compliant code.

Some authoring tools will allow you to create/designate a parallel version of the course that is 508 compliant, substituting text equivalent screens for animations and simulation objects. This version can be made available from the welcome screen.

A built-in compliance checker within the authoring tool can be useful, but you should also verify compliance by testing with screen reader software used by those with visual impairments and/or using an independent accessibility checker (see http://www.rnib.org.uk/xpedio/groups/public/documents/PublicWebsite/public_tools.hcsp).

For a complete summary of considerations related to authoring tools and Section 508, see the WC3's *Selecting and Using Authoring Tools for Web Accessibility* at http://www.w3.org/WAI/impl/software.html.

The WC3 also publishes an *Authoring Tool Accessibility Guidelines* document at http://www.w3.org/TR/ATAG10/.

There are tools such as the CourseAvenue Accessibility Player® that build Section 508 compliance into any online content.

For references and other information on Section 508 compliance, see http://www.section508.gov/

4.10.3. Aviation Industry CBT Consortium (AICC)

Support for this standard is common among authoring tools. The standard is widespread, has a long history, and not restricted to use within the aviation industry. See http://www.aicc.org/ for more information on this standard.

4.10.4. Standards for metadata

Some of the standards that are used specifically for metadata in eLearning are the following:

- IEEE Learning Object Metadata (LOM)
  http://www.imsglobal.org/specifications.html
- Dublin Core
  http://www.dublincore.org/

Support for a particular metadata standard in an authoring tool is not needed unless the standard has been fully adopted by your organization, or if you are think that the content will be reused by an organization that had adopted it. If the metadata standard has been adopted, authoring tool support for it can save you time in entering information that facilitates search, discovery, and cataloging of your eLearning and other content objects. In a large enterprise with many learning objects, this may represent a significant savings of time and effort. The authoring tool inserts the metadata fields into the content either into the assets or as separate XML files (in SCORM, it is the latter).

Note that SCORM does not prescribe use of metadata, or any particular metadata standard.

4.10.5. Common Cartridge

IMS Global Learning Consortium developed Common Cartridge as a standard way to package a course for importing to an LMS. It has many of the same advantages as the SCORM packaging standard (Content Aggregation Model). If you are developing courses that need to be packaged using this standard, you should look for authoring tool support to save you time and technical expertise. See http://www.imsglobal.org/cc/index.html for details.

4.10.6. Training and Learning Architecture (TLA) & Experience API (xAPI)

ADL has termed the next generation of SCORM as the Training and Learning Architecture (TLA). All current and planned future ADL technical projects, specifications and standards efforts fall within the scope of the TLA, an umbrella term that covers projects designed to create a rich environment for connected training and learning. Phase I of the TLA is focused on experience tracking that includes these four areas:

- A new runtime API (called the Experience API, or xAPI)
- A new data model
- A new data model format/syntax
- A new transport/communication method

The overall TLA vision also includes concepts for learner profiles, competencies, and intelligent content brokering to meet the needs for individualized learning content and systems. The TLA is not intended to replace SCORM, but SCORM, and multiple other types of content formats, will work in the TLA. The four components of the TLA are:

- Experience tracking
- Learner profile
- Content brokering
- Competency infrastructure

The Experience API or xAPI (formerly known as the 'Tin Can API'), the "experience tracking" component described above, is the farthest along in development currently (version 1.0 was released 4/26/13, and the spec is now at version 1.01). The Experience API tracks both formal and informal learning via 'streams' of learning experiences, similar to social media streams such as Twitter and Facebook. By capturing learning experiences via streams, learning can be mashed up with other activity data to fully analyze how it ties to performance. The new API enables the use of mobile devices, games, social networks, virtual worlds, and simulations in learning and training environments with the ability to track learning experiences consistently across devices and platforms. You could report that 'David watched a video,' 'David rated a video,' 'David tweeted a video,' and 'Jane retweeted David's video.'

Learning can also be tracked in real life situations and reported the same way. For example, 'John produced an audio track for a video,' 'Steven edited a video,' 'Ralph posted a video,' and 'Mary earned an Academy Award for a video.' This is why we describe this as "connected" learning, because even "real life" situations can be connected in more ways than just how people interact with computers on the Internet.

The xAPI is a specification that describes an interface and the storage/retrieval rules that developers can implement to create a learning experience tracking service. The service works by allowing statements of experience (typically learning experiences, but could be any experience) to be delivered to and stored securely in a Learning Record Store (LRS). Widespread adoption of the xAPI may drive LMSs to include an LRS component that can handle xAPI statements.

One major advantage of the xAPI over SCORM is that it does not require launching content from an LMS; in fact, it does not even require Internet connectivity while the user is engaged in the learning experience. Learners can connect after the fact to allow the xAPI to record their learning experiences. This has obvious implications for the future of LMSs; to accommodate learning that is developed for use outside of the LMS environment, or disconnected use, LMSs may need to separate their function that handles tracking of learner experiences into a single cloud-based service (in xAPI terms, an LRS) that is easily accessible from a variety of content and can dynamically capture xAPI statements describing learning experiences.

Authoring tools will need to be able to generate xAPI statements behind the scenes, without requiring the author to do manual coding. Any user action can theoretically be encapsulated in an xAPI statement and tracked by an LRS. The flexibility of an authoring tool in defining xAPI tracking nodes may become an important differentiator among authoring tools, once the standard is widely adopted.

For information on the xAPI standard, see http://adlnet.gov/adl-research/performance-tracking-analysis/experience-api/. There is a list of current adopters of the xAPI at http://adlnet.gov/adl-research/performance-tracking-analysis/experience-api/xapi-adopters/.

Currently, adopters of the xAPI are focused on offering the option of performing SCORM-like functionality using the xAPI instead of SCORM, rather than leveraging the unique features of the xAPI.

Lectora® is an exception; it has robust support for instrumenting eLearning modules with xAPI tracking of user actions and conditions (either canned or author-defined), including templates that automatically set up xAPI tracking for common objects such as quizzes.

Expect to see other vendors implement this support eventually, albeit with some rethinking of major portions of their authoring tool product model.

As of March, 2014, the following authoring tools supported xAPI to some degree:

- Adobe Captivate®
  http://www.adobe.com/products/captivate.html
- Articulate Storyline®
  http://www.articulate.com/products/storyline-overview.php
- Claro®
  http://classroom-aid.com/mobile-learning-authoring/
- Instancy®
  http://www.instancy.com/apis_and_integration.aspx
- iSpring Presenter®
  http://www.ispringsolutions.com/ispring-presenter/key-features.html
- Lectora®
  http://lectora.com/
- Knowledge Guru®
  http://www.theknowledgeguru.com/
- ZebraZapps®
  https://zebrazapps.com/

A list of xAPI adopters can also be found on the ADL Web site at http://adlnet.gov/adl-research/performance-tracking-analysis/experience-api/xapi-adopters/It is important to understand that these early implementations do not usually offer truly open-ended options for including xAPI statements. The interaction nodes available for including xAPI statements are generally very limited, and for very basic forms of data, such as assessment scores. In most cases xAPI is simply offered as a parallel alternative to SCORM for communicating key data about the learner's activities. However, expect to see much more flexibility soon in attaching xAPI statements to all sorts of interactions within authoring tools, due to growing demand for the affordances resulting from use of xAPI.

One advance in this area that may be coming soon is the ability of an authoring tool to ingest a web-accessible XML file storing an xAPI profile. This would automate configuring the authoring tool's ability to record tracking data that is relevant for a particular use case or community, such as the medical community, or for learners who are watching a video.

4.11. Assessments

ADL recommends that you create assessments within content so that they are portable and interoperable; however, in some cases, you may want to be able to create assessments through tools offered within the LMS rather than through the external authoring tool used to create content. Many LMSs offer this. The downside to using this internal LMS authoring function for assessments is that these assessments are often permanently resident in the LMS and cannot be exported for use in another system or with other content.

Assessment authoring within the LMS may be attractive because this ensures that assessments interwork closely with the LMS tracking database. It is often quicker and easier for LMS instructors and administrators to use an internal LMS function rather than create external assessments with the appropriate data calls. Also, assessment interactions can be more difficult to program than presentation content, so it avoids this technical burden on the authors as well.

Use of internal LMS assessment authoring is particularly common in cases where learning activities are conducted offline and cannot be assessed and tracked by the LMS while the student completes them. Thus, an LMS-delivered assessment is the only way to verify and store the student's level of mastery, and it is easier to author these assessments internally in the LMS.

For more information on LRSs, see the ADL white paper *Choosing an LRS* at http://adlnet.gov/adl-assets/uploads/2015/10/ChoosingAnLRS.docx.

4.12. Responsive design

Simply put, responsive design "…is the approach that suggests that design and development should respond to the user's behavior and environment based on screen size, platform, and orientation." (Lee, 2014). Originally, it was narrowly focused on making desktop eLearning suitable for display on mobile device screens. Now, it is more focused on tablet vs smartphone display issues, although laptop and desktop computers are still accounted for in most responsive design solutions.

It is important to emphasize here that it is not simply a matter of producing a "liquid layout" that automatically adapts to the screen size and orientation of the device. Most responsive design practitioners emphasize that many other differences in the way devices are used must be accounted for in true responsive design. For instance, tablets are normally used while sitting down, and smartphones are used while the user is standing and even moving about.

The basic premise of an authoring tool that achieves responsive design is that the authored content detects the screen size, platform, and orientation of the device, and through built-in scripts make the adjustments automatically to various parameters of the format for optimal user interaction with the content. This does not mean just shrinking or expanding elements on the page. It could involve moving locations of objects, and it could go so far as hiding information behind a "More…" link. The goal is to be able to author content once and have the content intelligently adapt to the user, not author for each different device.

Authoring tools such as Adobe Captivate 9.0 are now appearing that achieve various degrees of responsive design. There are significant differences in the ways that responsive design is implemented, however. Always check "under the hood" to ensure that you truly have the ability to author once for the many different scenarios you require.

There are currently three approaches authoring tool vendors are taking to responsive design. Probably the most popular is the approach exemplified by Adobe Captivate. This tool, like many other authoring tools, relies on a "slide" metaphor to present content, rather than a scrolling web page. There are "breakpoints" for each category of device; if a screen size is bigger than the size established for a given break point, the next larger breakpoint layout is displayed. Breakpoints can have master child relationships, or they can be authored independently.

Then there is the approach adopted by the open source tool Adapt, which uses a scrolling web page metaphor; content is authored as inline elements, in a "liquid layout". That is to say that when squeezed by a smaller screen, content will reflow down the page. Elements will not lose their relative position because they are positioned as part of the sequence of content objects.

Finally, Articulate Storyline has adopted an approach that relies on a mobile course player that responds to different screen sizes and orientations. It adjusts to smaller screens by doing such things as hiding sidebar menus and eliminating browser chrome. They also plan to include features in the authoring tool that allow web-based courses to run directly in modern web browsers without the need for a player.

5. List of possible requirements for authoring tools

The following is a comprehensive list of possible requirements for authoring tools you may be acquiring. It could also be used to assess the quality and suitability of authoring tools. The applicability of items in this list to your situation will probably vary widely; some items may be mission-critical for your organization and some may not be pertinent at all. You need to carefully weigh the importance of each in evaluating authoring tools. If you rate your list of tool candidates simply by all items in the list without weighting each item for its importance to you, it could skew the results, which could lead to a poor final choice for your system.

There is also the issue of the degree of support that an authoring tool provides for a certain feature. Very few of the features listed below are either 100% present or 100% absent in a given authoring tool; they may be stated in terms that are very measurable and specific or high level and subjective. You may need to translate the latter type of items into specific terms that can be easily assessed in a given product.

The following list of features is grouped into major areas. These are irrespective of category as defined in 3. *Categories and examples of authoring tools*. Some of these criteria may not apply to your situation.

It is important to remember the simple fact that most users, in many cases regardless of their skill set, will follow the path of least resistance in using an authoring tool, as with any other software. In other words, users will gravitate towards the most heavily optimized system features—those that are prominently available in the interface and easiest to manipulate. The system may include many advanced capabilities, or even easy workarounds or hacks that are possible, but most users will ignore these if they are not designed to follow the path of least resistance. This can drive a "dumbing down" of the sophistication and quality of the end product, reducing learning effectiveness to below desirable levels.

So the question is not necessarily, "What can the tool do?" but, "What can the tool do in a right-out-of-the-box, plug-and-play, easiest/most-obvious-path use case scenario?" Just because a vendor is able to make a technical case that their system has a particular capability doesn't mean that it is implemented in a way that is easy for users to see, understand, and use. Thus, when evaluating tools against these criteria, you should be looking to go beyond checking a box that indicates simply whether the tool has the feature or not, but evaluating how well optimized, emphasized, and implemented the feature is in the software, so that authors will be inclined to use it. This applies especially to features that you consider important to your training mission.

Before you even get to the point of asking "What can the tool do?" you should of course ask, "What is the learning goal I am trying to accomplish?" It is dangerous to allow the capabilities (especially those that are easiest to use, as mentioned above) and limitations of your authoring tool to dictate design. The learning goals and performance problems you are addressing should be clearly defined before choosing any particular tool; only after defining these should tools should be chosen, based on how well optimized they are for the particular learning outcomes you require.

As with most software, systems that are easier to learn and use have fewer capabilities, and vice versa (see 3. *Categories and examples of authoring tools*). Sophisticated capabilities will generally require a system that is harder to learn and/or require specialized professional expertise. It is important to determine the skill sets within your pool of authoring tool users, so that you know what you are prepared for and/or what you might have to acquire in terms of staffing or training. You can engineer your staff expertise and roles to match the out-of-the-box system, but it is usually not cost-efficient to engineer the system to match staff expertise.

This also applies to production task flow; you will almost invariably need to decide whether you want to change your internal processes to match the built-in authoring tool task flow, or vice versa (i.e., reengineer the authoring tool to match how your organization does things). Above all, do not underestimate the financial pressure you may find yourself under to tailor your organizational policies and processes to make it easiest to work with the authoring tool in its out-of-the-box implementation. Customization of authoring tools, whether open source or commercial products, can be quite expensive.

5.1. Criteria applicable to desktop and web-based tools

5.1.1. Support for instructional strategies and learning technologies

- Allows use of a wide variety of instructional strategies and learning technologies. For example:
    - Social media (see 7.13. *Support for social media*)
    - Mobile learning (see 3.4. *Mobile learning development tools*)
    - Immersive technologies (see 7.14. *Support for immersive learning technologies*). This includes:
        - Simulations
        - Serious games
        - Virtual worlds
- Supports the new ADL Training and Learning Architecture (TLA), allowing tracking of learning experiences in a wide variety of learning technology contexts. See 4.10.6. *Training and Learning Architecture (TLA).*
- Supports addition of game elements, like points, rewards, leaderboards, etc.

5.1.2. Sequencing and navigation

- Allows branching and sequencing depending on user responses to assessments and interaction results (ideally interoperably, using a standard like SCORM). This can take many forms, for instance:
    - Pretests that determine which parts of the course the learner can skip

  - Directing the learner to appropriate remediation after a post test
  - Branching to a different part of the course depending on a choice made in an interaction or assessment. Allows branches within branches.

- Allows custom learning paths based on user profile.
- Allows setting of global or default control modes for navigation between course elements—screens as well as units of content structure. For navigation between units of content structure, SCORM is recommended. In SCORM, the three control modes are:
  - **Flow -** user can only go to the next item in the sequence, and they can only go back to the previous item in the sequence (but can't jump back to any previous item they wish at any point in the course).
  - **Choice** – the learner can go to any item in the course at any time.
  - **Forward only** – user can only go to the next item in the sequence, and they cannot go back to any previous item in the sequence.

  In SCORM, these "items" are SCOs or aggregations (groups of SCOs), both of which usually consist of more than one screen. However, to set modes of navigation within SCOs (moving from one screen to the next), the authoring tool would need to implement the modes of navigation using proprietary features.
- Allows setting of bookmarks to preserve course location between sessions, either automatically upon exiting the course or through user action. In SCORM, the LMS stores a bookmark for the initial screen of the SCO the user was on when he or she exited. This SCO may be many screens in length, however. It would be up to the authoring tool to implement (ideally through the SCORM *cmi.location* data model, not through cookies) bookmarking of a specific screen within the SCO.

5.1.3. Assessment features

- Supports a sufficient number and flexibility of assessment types. Many LMSs provide an internal capability to create assessments (see 4.11. *Assessments*). Most tools have special templates and features for creating assessments. The standard types of eLearning assessments that you should look for are:
  - Multiple choice (both single and multiple answer)
  - Fill in the blank
  - Matching
  - Drag and drop
  - Ranking/Ordering
  - Image selection
  - Word scramble
  - Labeling an image
  - Essay or Short answer (requires instructor intervention to score answers)
- Questions include the ability to display a graphic or reference (e.g., PDF file) associated with the question and answer choices.
- Assessments can be randomized. This is common for assessments authored within the LMS, but can also be implemented in the content itself, and it is generally preferred to do it this way, since it makes the content more portable. Options include:

  - Questions are dynamically pulled from randomized bank of questions
  - Randomized bank pulls preset number of questions per objective
  - Order of questions is randomized
  - Order of question answer choices is randomized

- Includes built-in remediation features for assessments. There are two flavors of this feature: custom remediation for individual assessment items and remedial navigation to review existing content on appropriate course screen(s). The latter feature relies on being able to associate content screens to learning objectives and objectives to assessment items within the authoring tool. Options include:
  - Custom remediation for questions (regardless of wrong answer)
  - Custom remediation keyed to individual answer choices
  - Navigation to specific content screen(s) to review in preparation of retaking an assessment. This requires a jump function that sends the user to the required screens and then back directly to the assessment
  - Remediation can be configured as on or off
- Provides capability to include hints for incorrect answers on assessments—different for each wrong answer, and possibly for each wrong answer try.
- Allows you to configure the number of tries allowed for assessments (globally for the course, or individually by assessment).
- Can configure certain questions in an assessment to be required to be answered and others not required, and scoring based on weightings.
- Can configure assessments according to different types, such as pre-tests (where students can "test out" of the course), post-tests (required for passing the course), and self-assessments. Course behaviors can be set to be conditional on the results of these different types of assessments.
- Allows options for handling of test results (ADL recommends you implement this interoperably using SCORM)
  - Questions linked to objectives
  - Questions and possibly assessments are assigned weights
  - Passing threshold can be configured
  - Assessment questions can receive partial credit
  - Assessments can be timed, with option to time out learners who take too long
- Allows importing of external sets of questions in a standard format such as QTI.

5.1.4. Technical characteristics of output

- Provides a high level of support for standards such as SCORM (4th Edition is the current standard), xAPI, Section 508, and AICC, among others. See 4.10. *Standards support* for a description of the criteria for SCORM and Section 508. Some questions to ask in this regard are: does the tool include compliance checkers? Does the tool warn you if you try to do something that will make the course non-compliant?

- Supports many media file formats (especially formats that are used in your organization), as well as configuration of the media files within the tool. See 5.8 *Media handling*. For example, some tools can embed playback parameters and a controller interface for a video into the web page. Note that many authoring tools cannot produce simulations requiring complex variable manipulation; you need a specialized tool for that.
- Uses the multitouch technology embedded in modern user interfaces (especially mobile) to allow learners to tap, scroll, pinch, and swipe within eLearning.
- Enables automatic scrolling of content on mobile devices that may have been originally designed for a larger screen factor.
- Produces output supported by a wide variety of platforms (e.g., Mac, PC), browsers (e.g., Internet Explorer, Chrome, Mozilla, Firefox) and older versions of them, and screen sizes (e.g., smartphone, tablet, desktop)
- Supports High DPI displays, like Apple's retina display.
- Enables detection of device and delivers the appropriate course version, even if the learner changes from one device to another while working their way through the course.
- Supports responsive design. This allows you to create a single project, not a different version for each device. The content dynamically adapts itself to tablets, mobile phones, and desktop computers. This implies options such as:
    - Being able to choose the portion of an image you wish to display on a smaller screen.
    - Out-of-the-box themes to render differentiated content for multiple devices
    - Fix the minimum and maximum size of objects so that they do not rescale across device views.
- Includes pre-built modules/scripts that can easily be inserted to check learner systems for necessary plug-ins, etc.
- Allows insertion of web pages and other web content into containers (such as Webobjects) that can be displayed within authored content.
- Imports native HTML5 animations without requiring any plug-ins.
- Supports output to mobile devices (see 4.2. *mLearning authoring tools*). Some tools offer this as an output option, though they may not be designed specifically for this type of learning.
- Requires a minimum of players and plug-ins, especially proprietary ones that are not automatically installed with the browser.
- Supports foreign character sets (Unicode or other multi-byte fonts), especially Asian characters.
- Supports creation of a desktop executable file that can run on CDs or DVDs or run on the desktop after being downloaded from the intranet when there is limited bandwidth.
- Supports insertion of custom scripts that are produced outside of the tool
- Supports integration of content with mobile device capabilities such as geolocation

- Output files can be packaged as hybrid mobile apps and distributed through Apple Store, Google Play, etc.
- Ideally, has an output file format that is identical to the internal source file format, and that this format is clean, universal code, ideally HTML5; none or a minimal amount of proprietary code is involved. This allows courses to be imported and edited in other authoring tools, and greatly enhances the durability of the output. Some authoring tools rely on special code inserted into HTML comments fields, or Java applets, for instance, to implement certain functionalities. If the authoring tool is no longer available, it may be difficult to interpret and edit this code.
- Has options for enabling and configuring printing of the course screens. This enables authors to view the course in a storyboard type of format (ideally, in an editable word processing document) and enables students to print sets of course screens (ideally in non-editable PDF format), rather than limiting both groups to using the browser Print function (which will only print the current screen).
- Has options for export as a screencam movie file for offline and mobile device viewing. Of course, this will strip out any interactivity. If important content is hidden behind popups, etc., you would not want to use this feature.
- Output works well with Section 508 accessibility technologies like screen readers.

5.2. Authoring of documents related to course

- Includes templates for authoring of Glossaries whose contents can be automatically linked to "hotwords" in the course.
- Allows templates for authoring of course FAQs.
- Includes templates for authoring of course tutorial and Help pages.
- Includes templates for reference and resource pages (possibly in Word, for conversion to PDF).
- Provides support for creating student surveys, certificates, and course evaluations. Many LMSs do this, but you may want this feature in your authoring tool so that these items are portable to other LMSs.

5.3. Ease of learning and use

- Is easy to learn and use, ideally with the ability for users to choose from tiers of features according to the knowledge and expertise of the user. This allows users to start using the program quickly and gradually progress to more complex authoring tiers/feature sets as their skills mature. In other words, users only see features that are relevant to their level of skill and the kind of operations they are capable of performing.
- Displays interfaces that are consistent and standardized throughout all screens.
- Allows customizing of workspaces and saving them
- Includes grids that allow convenient layout out of objects, and snapping to the grid and other objects
- Has robust panel management features (docking, stacking, etc.)

- Uses straightforward, simple, and intuitive paths for performing authoring functions. You should test your most common and important use cases on the system to verify this.
- Provides user interface customization (not on the level of tiers of features, as above, but on an individual feature basis), so that users can optimize for their particular needs.
- Is easy to install and configure, ideally not requiring a system administrator (possibly using wizards).
- Includes spell check and grammar check features
- Provides clear, specific error messages and diagnostics that aid in troubleshooting. A generic message that is the same for all errors is not acceptable. You also want to avoid cryptic, technical messages that can only be interpreted by the tool's software developers. Messages should be understandable not just be technically-inclined tool administrators, but also content developers.

5.4. User training, support, and documentation

- Has robust support documentation in a wide variety of forms including tutorials, help, examples, references, and user manuals.
- Has a variety of Help Desk support options for administrators and learners (telephone, chat, email, etc.).
- Has a Help Desk system that is structured and process-driven via trouble call tracking and reporting.
- Has Help Desk support that coordinates problem resolution with the appropriate parties: vendors, SME's, etc. for problem resolution.
- Has knowledgeable, experienced support personnel.
- Is available as close to 24/7 and world-wide as possible.
- Offers extensive training options: eLearning, video tutorials, ILT sessions, webinars, etc..
- Has onsite training options. If training is at vendor site, the location(s) are a reasonable distance.
- Includes an orientation tutorial for new users.
- Has free online forums for support.
- Has an established user community to turn to for help (this is generally true only of the tools that are in the most widespread use)
- Has a low average turn-around time for help-desk support.
- Has a feedback function for suggestions on improving the product.
- Provides technical consulting services options for customizations, implementation, configuration, architecture design, needs analysis, change management services, etc.

5.5. Technical architecture

- Does not arbitrarily limit the number of levels, objects, or sizes of items included in the course.
- Supports a wide variety of delivery architectures. For instance, if you have an eLearning architecture involving a dedicated content repository (that may be on a different server than the course delivery system), the tool supports configuring the eLearning for this.
- Has reasonable system requirements that are attainable within your organization (both for authors and learners).
- Has the ability to call external applications and code objects (such as calculators and random number generators), and set up interfaces to read and write from databases.
- Uses an open architecture that allows additional functionality to be added from external sources.
- Is interoperable with other authoring tools, based on input and output file formats, etc.

5.6. Acquisition and maintenance

- Has a licensing agreement that is flexible and easily scalable to reflect changing number of users.
- Costs less than competing authoring systems with the same or similar feature set. This includes all TCO (total cost of ownership) costs.
- Costs less for recurring and ongoing support compared to the cost of other similar systems.
- Is projected to cost less for required customizations compared to the cost of customizations for other similar systems.
- Costs less for add-ons such as APIs to external applications compared to the cost of other similar systems.
- Offers a subscription pricing model that is reasonably priced without requiring an excessively long-term commitment/contract up front in order to subscribe

5.7. Automation and process optimization

- Includes a convenient mechanism for adding metadata or descriptive labeling to course components (for SCORM courses, this should include the ability to attach metadata at the course, aggregation, SCO, and asset levels). Ensure that the metadata format is the one that your organization uses.
- Integrates the storyboarding process into the tool. Some tools integrate storyboarding with the authoring process, so that most files can be output when the storyboard is complete, without any need for further production. Some tools support using PowerPoint to create storyboards, which can be imported, edited, and turned into eLearning.
- Allows searching within individual courses, and across course libraries, both while authoring and after it is published on the LMS (the latter may only be available as an LMS feature, depending on your implementation).

- Includes options for automating the creation of course navigation functions from the content. For example, creating course maps, menus, and table of contents from screen titles, or keyword glossaries from identified hotwords.
- Stores links in an externalized database file so they can be updated from a master instance.
- Allows importing content in other file formats (especially Microsoft Office®).
- Has mapping feature that allows you to indicate how the styles and items in Microsoft Office documents to be imported relate to how it will be inserted into the course. For example, an "H1" heading in a Microsoft Word document becomes a screen title in the LCMS content object.
- Allows importing of content packages such that they are fully editable (for SCORM content).
- Provides features that allow authors to view the course structure in a graphical representation (diagrams, outlines, etc.) using a variety of metaphors, for example, object trees and flowcharts. It should allow not only viewing but creating and editing course structures with navigation links to nonexistent or placeholder screens before these screens are populated with content. Reorganizing and reordering lessons and screens should be as easy as dragging and dropping structural elements.
- Includes a spelling and grammar checker.
- Has robust support for building tables and diagrams.
- Has global find and replace function.
- Is optimized for reusability in general (not just measured by SCORM support). Some tools have their own internal content repository that allows mixing and matching objects, allowing you to pool assets in a media library so they are reusable across courses authored with the tool. This feature is common for LCMSs, but not for other types of authoring tools.
- Includes a wide variety and numbers of templates or skins that you can use out of the box with little or no modification, and supports easy creation and application of new templates and skins. It should allow templates and skins to be applied to any level of course structure (a screen, lesson, module, or the entire course).
- Allows authors to easily override elements (for individual screens) of the course-wide or screen templates and skins, to allow flexibility and creativity.
- Apart from templates, incorporates a library of reusable components (scripts, images, text pieces, etc.). If the elements in this library are updated or changed, these changes should propagate throughout the course. And they should be sharable across courses, not just within the same course.
- Incorporates some degree of ability to edit raw material assets at a low application level; for example, support for editing of images to the degree that use of image editing software like Photoshop® might not even be necessary.
- Has a high degree of traceability for the impact of all changes across all courses and objects (this applies mostly to LCMSs). For example, properties and attendant warnings that object x is used in courses y and z; changes to that object will affect those courses, and vice versa.

5.8. Media handling

- Audio
  - Allows embedding or linking to audio files.
  - Allows native editing of audio files "in place," including setting compression type and amount, and audio quality.
  - Allows author to record narration audio that is synched automatically to screens and screen sequences as recording progresses.
  - Enables set audio clips to launch based on user actions.
  - Allows providing closed captioning.

- Graphics
  - Allows native editing of images "in place," including applying special effects like transitions.
  - Allows setting graphics to be interactive elements that the user can drag and drop, link from, etc.
  - Internally and automatically converts imported graphics to standard file types used by the authoring tool (PNG or JPG).
  - Supports a wide variety of media (see below) and media file formats. Examples include:
    - Audio
      - MP3
      - RealAudio
      - WAV
    - Video
      - MPEG-4
      - RealVideo
      - Quicktime
      - AVI
      - H.264
    - Documents
      - Microsoft Office
      - Adobe PDF
      - HTML5
    - Graphics
      - JPEG
      - PNG
      - GIF
      - SVG
    - 2D animation
      - SWF
      - HLA Simulations
      - HTML5
    - 3D animation
      - SWF
      - WebGL

- Animation
  - Includes a library of animation objects, including characters/avatars.
  - Has the ability to build and/or incorporate HTML 5 animations.

- Video

- Allows embedding or linking to video files.
- Allows native setting parameters of video files "in place," including setting compression type and amount, and video quality.
- Allows author to record screen video (i.e., for system simulations).
- Enables set audio clips to launch based on user actions.
- Enables annotating videos with timed overlays.
- Allows providing closed captioning.

5.9. Programming features

- Provides browser emulation (or previewing in a separate browser window) that allows quick previewing of screens and objects exactly as they appear and function in the target browser.
- Includes revision tracking to audit changes and roll back to earlier versions.
- Runs validation checks of HTML code.
- Incorporates "round-trip" editing of code, meaning that changes made while in code editing mode (for example, directly making changes to the HTML code) are immediately reflected in WYSIWYG editing mode, and vice versa. This also means that WYSIWYG authoring functions do not overwrite or are not incompatible with HTML code entered manually.
- If the tool incorporates use of XML as the core internal format or as ancillary data storage (see 7.2 *Use of XML or JSON*), it has an XML editor that allows programmers to edit the XML directly.
- Provides the ability to launch source object editing applications from within the tool. For example, the ability to launch and edit graphic objects in Photoshop from within the tool; saving changes automatically updates the file in the target format (like JPEG).
- Allows authors to establish and control course file directory structures without rigid constraints. For example, it should allow authors to specify which assets are stored in which directory, and they should be able to easily rename and reorganize this directory later, updating links to associated files.
- Offers a scripting language (such as Adobe Flash®, ActionScript®, or Javascript) to extend the tool's functionality, with the ability to create and manipulate variables that control a wide variety of functions and behavior.
- Offers convenient features related to media handling, including:
    - Cropping of graphics, not just resizing
    - Adding alt-tag data for Section 508 compliance
    - Vector graphic creation tools (ability to create lines and simple shapes to aid in layouts)
- Ability to set control parameters for media objects (for example, Flash® animations) within the tool rather than requiring them to be set within the media object authoring tool. This includes looping behavior, streaming parameters, etc.

5.10. Criteria specific only to web-based tools

The following criteria apply only to web-based tools, and not desktop tools. These criteria are added to the list above.

5.10.1. Collaborative authoring and process management

- Offers "organization aware" features that allow collaborative server-based authoring based on organization roles and permissions. Permissions should be able to be assigned not just by organization or role, but course or project as well.
- Includes project management features, to help project managers plan and track progress on individual screens and other components.
- Manages the production process efficiently. This may include built-in workflows (for approvals, for instance) and production, QA, and review pipelines. It is ideal for notifications (for example, telling the next person in the next role in the pipeline that the course is ready for them to work on) to be handled through email, not just an internal authoring tool notification mechanism (since you may not be able to depend on people logging in to the tool regularly to check their notifications).
- Includes configuration management and version control features such as checking files in and out to prevent accidental overwriting.
- Allows adding of media-empty placeholders with properties and tasks associated with them (i.e., requests for further action by other developers).
- Provides the ability to annotate and communicate actions taken, approvals, errors, etc. in regards to screens and content objects, for future reference or for other authors, using built in fields as well as email.
- Provides a means to assign tasks that are tracked and managed in the system, at the level of particular content objects (provides a means for scheduling content maintenance and management)
- When reviewers make comments they can make edits directly in the content (similar to Revision Tracking in Microsoft Word®).

5.10.2. System access

- Uses robust security architecture to maintain system access.
- Allows users to self-register or create a request for an account.
- Provides a single sign-on, so that users who have logged in to the enterprise intranet (through a portal, etc.) can get into the tool without additional login.
- Requires user logon only once per tool session.
- Uses Common Access Card (CAC) access (for high-security government installations).
- Incorporates appropriate security certifications and standards, and features (see 4.6. *Security considerations*). Other security standards you may need include SSL, PKI, and FIPS – 140-1.

5.10.3. System performance

- Performs with minimal latency under a variety of use case scenarios and load conditions.
- Handles reasonably large numbers of concurrent users.
- Handles user load efficiently, provisioning and scaling resources to smoothly accommodate fluctuations (especially spikes) in numbers of concurrent users.
- Works equally well (all functions, including especially course previews) on all standard Internet browsers.

5.10.4. Permissions and roles

- Defines a wide variety of permission and role levels that are applicable to a range of organizational structures and use case scenarios for the tool.
- Uses administration templates to easily set group permissions.
- Restricts access to authoring functions for individual or groups of courses based on membership on teams associated with those course(s).
- Allows delegating permissions for users at a lower level of permission than what one is logged in as.
- Allows creation of subgroups that inherit permissions of parent groups.
- Allows administration based on external data feeds concerning organization roles and permissions.
- Supports mirroring an organization's structure in the database to manage authors, content administrators/owners, programmers, and approvers based on where they exist within the organizational structure.
- Manages the administration process efficiently with built-in workflows (for approvals, for instance).
- Provides features that allow administrators to view role structures in a graphical representation (diagrams, outlines, etc.).
- Has administrative interfaces are clear, simple, and optimized for usability. Administrator interfaces are no less important than author interfaces. Just because author interfaces are well-designed does not mean the administrative interfaces will be also. This is particularly important where there is a need for non-technical staff to perform administrative functions.

6. General recommandations

- Some authoring tools (especially RAD tools) are relatively agnostic of learning theories and approaches. They are open-ended and flexible, and can be used to support many different types of learning (for example, scenario-based learning). However, many tools (especially ones that rely on use of templates) have an explicit or implicit assumption of the type of learning that will be produced using the tool, and are optimized to build that particular type of learning. Be sure to

determine what these assumptions are, and either look for a tool that supports it, or at least avoid tools that are optimized for a different, incompatible, learning type.

Do not underestimate the degree to which a tool assumes a certain type of learning. Most tools implicitly assume traditional declarative, cognitivist learning, especially in terms of the assessments they produce (standard multiple choice tests). Tools that are optimized for alternative learning types, such as constructivist learning environments, are not common. It is generally the case currently that you would need to use an open-ended tool to create for these learning types.

- Keep in mind that most software tools that are easier to learn and use have fewer capabilities, and vice versa. Sophisticated media and/or learning strategies will inherently require a tool that is harder to learn and/or require specialized professional expertise.
- If you do not have a designated, experienced programmer with a training background who develops your eLearning, it is generally better to predicate your choice of authoring tool on having an instructional designer learn to use it, rather than an IT person. In this case, a non-technical tool is better. All other things being equal, production will be faster, easier, and you will get a better quality product if instructional designers are doing the authoring.
- Avoid the first release of a new authoring tool.
- Ask the vendor who their other clients are, what they use the tool for, and see if you can talk to these clients about their experience using the tool.
- Ask the vendor for a demonstration in your facility, running your content on your enterprise system(s). The vendor may present a canned demo of the product on their system, and that is fine as a general overview of the tool's capabilities, but you should see how well the tool expresses these capabilities within your IT environment.
- Avoid authoring tools created by companies that have a short history in the market (less than 5 years), or have been operating for a short time, or have a small organization. You also want to watch out for companies that are about to be acquired or merged with another vendor.
- You might want to check to see if the company is International Standards Organization (ISO) and/or Capability Maturity Model Integration (CMMI) certified to ensure the quality of their software.
- Determine exactly what capabilities you really need. If you already have a course delivery system, for instance, you may not need that capability that is included in an LCMS. Many LCMS vendors sell the authoring module as a separate application for a lower price.
- For a web-based tool, determine whether a hosted solution may be right for you (see 4.4. *Hosted solutions* for more information). In most cases, outright ownership is the best route. However, a hosted solution may be cheaper in the long run, in terms of server usage and saving the hassle of performing your own admin and maintenance functions.
- Do not overlook open source, freeware, or GOTS solutions; solutions may be available at very low overall cost that adequately meet your needs (see 4.3. *Open source, freeware, and GOTS solutions* for more information).
- As described in 1. *Purpose and scope of this paper*, assume that you will need several authoring tools in combination; a primary one for authoring the "shell", and secondary/auxiliary authoring tools that are optimized for particular capabilities or assets. This is very commonly done in the

case of courses that are authored in DHTML (for instance, using Adobe Dreamweaver®, with Adobe Flash® objects inserted for animations).

- Consider the current roles, responsibilities, and skill levels of the people who will do authoring, and how much you are willing to ask them to learn new skills and change the parameters of their job to become tool experts and take on the role of authoring, if they are not doing authoring now. A simpler, less powerful tool may be the best option in order to avoid having to make significant changes in your personnel landscape.

  This also relates to the question of whether your authors or authors-to-be are generalists or specialists, and whether it is realistic or desirable to force them to become more of one or the other. Tools that are simpler and less powerful will be better suited to those who want to remain generalists. Those who are currently generalists will be resistant to the technical nature and steep learning curve of a complex tool. For instance, an instructional designer who is also a course developer (i.e., generalist) may use a simple tool for development that allows him or her to spend most of their time on instructional design, rather than wrestle with the technical nuances of a complex and powerful tool as a specialist developer).

- It is generally better to make a more powerful and flexible program work for you via carefully designed, robust templates than to use a less powerful tool that owes its ease of use to limiting what you can do. If you set up your templates and workflows for using them correctly, the learning curve and level of effort of the more powerful tool will eventually be on a par with the less powerful, easier to use tool—but you will always be able to call on the added power and flexibility of the more powerful tool if you need it.

- Try the tool out on the system configuration your authors would typically use in your training organization. You may discover some surprises in performance and features that you would not otherwise have found. For instance, the authoring tool's preview function may actually preview screens quite differently than what they look like in the actual end-user browser.

- Determine the skill sets within your pool of course authoring staff, so that you know what you are prepared for and/or what you might have to acquire in terms of staffing or training.

7. Current trends in authoring tools

7.1. Team-based life cycle production and maintenance

Life cycle production and maintenance of courseware includes all of the phases of an eLearning project in a single tool's capabilities: analysis, design, development, implementation, and evaluation (ADDIE). In order for an authoring tool to support this, it must allow collaborative authoring and permission-driven production pipelines. This trend is driving many desktop tools to move permanently to web-based architecture, or at least to have a web-based option, since this enables all kinds of "organization aware" workflows (enforcing who does what when during production—centralized control with distributed contribution). More and more, tools allow modeling of organization structures and processes, and assignment of specific roles in the production process. These roles allow the tool to encompass a greater scope of production and maintenance activities within the ADDIE model, such as analysis and evaluation.

7.2. Use of XML or JSON

Like the software arena in general, tools are moving towards use of XML (Extensible Markup Language) or JSON (Javascript Object Notation) as an output format and/or as the internal authoring source code.

XML is a universal, durable markup language that is relatively easy to learn and use, and is a robust means for storing structured data. Because of these characteristics, some training organizations are requiring that their learning content be stored in its raw form in this format. Using a transformation application, XML stored in this format can then be output into many different formats, including all kinds of documentation that is not related to eLearning.

JSON is not a markup language, but functions similar to XML in that it is a universal data interchange file format. It has advantages over XML in that:

- It can be directly read in and "understood" by browsers, since it is part of the Javascript specification. XML requires some sort of transformation code or middleware in order to be understood by browsers.
- Its syntax is significantly simpler than XML.
- It is more compact than XML.

Similarly, tools are starting to appear that use XML or JSON as the means of storing the authored content internally. This XML or JSON content can then be compiled into an eLearning runtime file or set of files using, again, a transformation application. This "open architecture" approach achieves three goals:

- It separates content from appearance (see 7.3. *Separation of content, appearance*), which promotes greater flexibility in content maintenance, and more delivery options.
- In line with the term "open architecture," using an open, universal format (XML or JSON) for storing content allows the possibility of using that content in different output formats, applications, and contexts, depending on the transformation engine used, which could be a COTS (commercial off-the-shelf) application or custom one. In other words, use of XML or JSON takes the content data out of proprietary code objects and puts it into a universal file format, increasing the interoperability of this data with other (proprietary) applications and systems. For instance, some authoring tools (such as Flash) have the ability to read in data from external XML or JSON files, either at runtime or during the authoring process.
- In addition to allowing a variety of output formats as described above, it can free the authoring capabilities (which determine the complexity of learning interactions) from the typical constraints of feature sets presented by authoring tools with "canned" (i.e., non-scripted) feature sets. Custom scripts and code can be written to manipulate the stored content in various creative and complex ways, without interfering with the content itself.

One technically difficult feat for an authoring tool is to import courses created in other authoring tools such that they are fully editable. Unless the course is 100% free of proprietary code, it may be difficult or impossible for the tool that is importing these externally-created courses to understand and interpret this code. One solution is to use XML or JSON, which avoids using proprietary code for storing the content (the modules that transforms the content may however be proprietary), thus making importing and editing content between tools more interoperable. This relies on tools being able to import XML or JSON files. It would then be up to the authoring tool to apply the correct transform to the XML or JSON data to output into screens.

Though the content data may not be imported into the tool in the form of XML or JSON files, and XML or JSON may not be used internally as the means for storing content, the tool may still have an option of compiling the content portion of it into XML or JSON output. This XML or JSON output can be used as part of the runtime file set (i.e., data is read in from it by the runtime engine to assemble screens), or it can be imported into other tools or used to create other formats, as described above.

A further advantage of using XML or JSON is that it enables direct editing of content in a text editor or—through using an XML or JSON interface application—a web form in order to update content, making the content updating process simpler. In this case, the content updaters (who may be SMEs or instructors) can

make changes to text, change URLs, etc., without needing access to or experience with the primary authoring tool.

7.3. Separation of content, appearance, and function

Separation of content, appearance, and function in content authoring is now firmly established in web-based training. It is usually implemented by writing content with HTML, controlling its format with CSS, and creating interface behaviors and user input processing with Javascript. Being able to control these three things independently has huge advantages. It allows content authors to apply styles to content (similar to styles in Word®) in order to flexibly make changes to all screen layouts and objects at once. This is especially important for responsive design for mobile learning. Putting functions on another separate layer allows screen behaviors to be changed in one place (a configuration or script file) across all screens as well. In general, separation of content, appearance, and function facilitates flexible updating of text, media files, etc. without recoding the screens they appear on.

There are also self-contained tools that do this as well. Examples include Cold Fusion® and server technologies like ASP®. This trend has permeated deeper and deeper into application architecture. Some tools (and systems like LCMSs) that use this principle rely on dynamic assembly of eLearning at runtime (which requires server software); others assemble and solidify the final product at the time files are published (handled within the authoring tool). XML or JSON are common means for storing the "content" portion of the equation (see 7.2. *Use of XML or JSON).*

Another way that separation of content and appearance manifests is the separation of the course interface from the content. Most often this involves use of skins, which are interface designs (possibly including functionality as well as visual design) that can be swapped out easily.

7.4. Support for ISD Process

Some tools are adding support for the ISD process—in other words, the activities that led up to (and possibly come after) the design of the course that is rendered in the authoring tool. This usually includes wizards, coaches, and templates, as well as checklists for doing training needs analysis, writing design documents, determining instructional strategies, writing learning objectives, etc. This ISD support is often targeted at non-instructional designers, i.e., SMEs who know little or nothing about instructional design.

7.5. Integration and complexity of templates and skins

Templates and skins were discussed in 4.5. *Templates, themes, and skins.* Templates and skins have always been a part of authoring tools, but they are becoming much more integral to the tools, and are becoming more complex. The tools to build and manage templates are also becoming more complex to keep pace with the templates themselves. This trend has the overall effect of simplifying the authoring process, so that the author only needs to focus on the information to be presented and instructional strategy, rather than format and function (which are automatically taken care of by the template).

7.6. Learning object-centric architecture

Authoring systems that are integrated with LCMSs or content repositories best exemplify this principle. They give developers the flexibility to develop all kinds of content objects (not just explicitly designed for learning purposes) and assemble and reassemble them in different combinations (often relying on SCORM to do this) for learning modules either at runtime or when courses are published. This trend reflects the growing popularity and movement towards knowledge management practices.

7.7. Embedded best practice design principles

Tools are integrating visual and instructional design principles more and more, as these principles are more accepted and standardized, and become the default working principles for eLearning.

7.8. Automated metadata generation/extraction

Tools are making the onerous task of determining and entering metadata (particularly for SCORM courses) easier by extracting directly or intelligently inferring (using latent semantic analysis [LSA] technologies) data for certain metadata fields such as keywords, learning time, reading level, etc.

7.9. Open architectures

"Open architecture" infers that the tool has APIs that allow integration of external applications and systems into the tool, including, in some cases, swapping a tool vendor-provided function with an externally produced one. Open architectures imply a relaxation of proprietary control and constraints on the part of the tool vendor, allowing potential users to "look under the hood" at their implementation.

To enable open architecture, the vendor usually must share all or parts of its architecture with add-on/system integration developers. This may require some license agreements between entities sharing the architecture information.

In spite of the potential for competitive disadvantages resulting from publicly exposing the inner workings of their system, some vendors favor them because their customers want to be able to easily customize the system by purchasing additions that the tool vendor may not feel are important enough to develop themselves.

Open architectures have driven the creation of a marketplace for third-party applications that can be integrated into the core tool as modules. These modules can provide all sorts of functions, mostly revolving around advanced types of interactions and assessments.

7.10. Support for team-based learning

True team-based learning implies more than a group of learners in a meeting room taking a course together under one login, presenting themselves to the LMS as if they are one learner and making group decisions about how to complete course activities, or synchronously progressing through a course from different locations and being scored by the average of their individual scores. Team-based learning revolves around the idea of learning activities that both affect other team members' activities and are affected in turn by the actions of others in their team, who may be using a different version or part of the course based on their individual role in the team.

Thus, authoring tool support for team-based learning involves more than just providing communication functions in the content in order to provide collaboration and peer review by multiple learners. Complicated assessment and sequencing paradigms must be possible, with intelligent agents or middleware automatically tracking and mediating the activities and performance of each team member, and reporting rollup progress as well as an audit trail for how these scores were generated (based on individuals' performance) to the LMS.

The technological challenges in this type of learning are now being worked out, but there is no universally accepted solution, so no prominent authoring solutions to support it have appeared yet. But as soon as the team-based learning paradigm becomes an established part of the training and education space, authoring tool and LMSs will surely move to support it.

7.11. "Gadget"-based interface

Gadgets (aka "widgets" or "applets") are functionalities that are presented as separate items on a page. They are used in many commercial e-mail "My Page" interfaces, and in many enterprise portal interfaces. They make it possible to completely customize the user interface; gadgets can be turned off so they do not appear on the interface, and can be moved to any location on the page. They can be associated with a specific role so that users only see the ones that are relevant or permitted for their role.

This type of portal-like interface has gained traction with some vendors, simply because users are more comfortable with this type of modern interface, and it allows a high degree of interface tailoring to suit their needs.

7.12. Interactive images

Interactive images refer to eLearning screens that are filled completely with a single graphic. Areas of the graphic that infer an available "bucket" of content are linked to further information or media. An initial click zooms in to that area of the screen (delineated by borders or an object). Clickable or rollover areas within this area are then activated to provide some kind of detailed information. These screens can be a welcome relief from otherwise text-centric screens, and provide a modicum of interaction.

Interactive images are not difficult to create in most authoring tools, and templates specifically for this kind of object are available. For templates and a tutorial in how to create one, see http://blogs.articulate.com/rapid-elearning/how-to-create-an-interactive-image-template/?mkt_tok=3RkMMJWWfF9wsRoluaXPZKXonjHpfsX97eksW6K0lMI%2F0ER3fOvrPUfGjI4DRMpmI%2BSLDwEYGJlv6SgFT7jGMblo27gPWxA%3D

7.13. Support for social media

Learning experiences are now being designed to include elements outside of the traditional didactic eLearning model. They often involve user-generated, and decentralized sources. These elements are generally termed "social media." The list of types includes:

- Wikis (for example, Wikipedia)
- Social networking (for example, Facebook®)
- Blogs (for example, Blogger®)
- Micro-blogs (for example, Twitter®)
- Social bookmarking (for example, Delicious®)
- Social news (for example, Digg®)
- Picture sharing (for example, Flickr®)
- Video sharing (for example, YouTube®)
- Communities of practice (CoPs)

Courses can be authored to include these elements, as APIs; a learner could, for example, be given an assignment to research a topic in some of these tools. The API would embed the functionality into the content as a "widget" on a course screen. However, access to these social media elements is usually not provided within the content, but rather, the course author configures the LMS to provide the access to the social media site or function through the LMS interface.

A recent emerging trend in social media-based courses are "massive open online courses" (MOOCs). These are courses where both participants and course materials are distributed across the Internet. They are usually based on informal learning principles, relying heavily on social media. Learners participate at their own level of time and interest, and there is no cost. Universities are usually the sponsors of MOOCs. Rather than author and deliver original content, you may be able to leverage content or curriculum

components that are already offered in an MOOC. For more information on MOOCs, see http://en.wikipedia.org/wiki/Mooc.

7.14. Support for immersive learning technologies

There is growing interest in developing learning for serious games and virtual worlds. Tools are now appearing to support developing learning experiences for these, although the authoring paradigms are very different in the sense that you are not authoring screens as in an eLearning course; you are creating 3D environments that have particular interaction nodes, and, in the case of games, a narrative that drives the sequence of activities, as well as competitive and incentivizing elements such as rewards, points, and leaderboards.

With these technologies, authors do not create course packages and learning objects that can be uploaded to and delivered from an LMS. They require special players and extensive server software to enable them. Most virtual worlds require development to take place inside the environment itself. Assets (3D objects) can usually be created inside or outside of the virtual world, but assembling the assets into a learning scenario requires tools and techniques within the platform.

Most virtual world learning implementations involve synchronous learning exercises using live avatars. Asynchronous implementations are currently mostly just rendering of traditional 2D eLearning through a web browser either inside of the virtual world or in a daughter window of the virtual world application. This type of implementation can be created using traditional eLearning authoring tools. Asynchronous multiplayer implementations involve “bots” (scripted avatars that operate autonomously). These are slowly appearing in learning implementations, but are technically advanced to develop and implement.

For synchronous learning experiences in virtual worlds and games, the authored “course” consists of three parts:

1. Building out the environment in which participants are to learn in.
2. Scripts for the avatars who take part in the learning exercise.
3. Assignments or challenges for the learner avatars.

A combination of authoring tools is needed to create all three parts. These tools are usually platform-specific, and offered only by the platform vendor.

7.15. Support for online assessment of performance tasks

With the growth and acceptance of informal learning approaches, there has been a growing consensus among educators and trainers that assessment needs to focus more on observation of student target performance and products thereof. The trend is away from relying on traditional multiple choice tests, whose relationship to the target performance may be tenuous at best.

Until recently, there was not much attention paid to designing special authoring software to create these types of assessments, since all this type of assessment really required was thoughtfully-prepared product/project assignments and evaluation rubrics. However, this area is starting to become systematized, formalized, and standardized via specialized authoring software, particularly in the K-12 education arena. An example of this is the Acuity Performance Task System® (see http://thejournal.com/articles/2012/11/14/new-acuity-tool-tackles-online-assessment-of-performance-tasks.aspx?m=1).

These systems allow users to create and assign performance tasks that mimic the complexity of real-world situations and draw upon interdisciplinary knowledge. The tasks are instructional tools as well as assessment vehicles. Scoring can focus on overall product/project performance as well as individual tasks

involved within a performance product/project. In K-12 education, these authoring tools include a library of common performance task scenarios in English, math, science, etc.

7.16. Support for semantic web/Web 3.0 technologies

Tozman (2012) argues that both online and offline learning presented as a formal event that requires some form of attendance (i.e., away from one's current tasks) is a dying breed. It is being replaced (rightly so, he says) by the just-in-time, just-in-place performance support paradigm.

Tozman says that the advent of semantic web/Web 3.0 technology (as exemplified in web sites such as Wolfram/Alpha and Open Cyc) will revolutionize learning such that appropriate content and curricula will be generated on-the spot, in accordance with the performance needs of the user at the moment of need. Semantic web technologies will apply human-like reasoning and ontologies of meaning to directly answer factual questions and recommend the correct action or decision. Event-driven learning may not entirely disappear; it could remain as one of many performance support options.

To sync up with this trend, he says, authoring tools will need to produce content in a form that is consistent with the evolution of web technologies like semantic web/Web 3.0. The content must be transparent and structured (for example, with XML) to allow semantic processing engines to understand its meaning and utility to learners. The authoring tool would not be designed to assign any specifics to the packaging and formatting of content at the outset; it would be designed to set up the rules for semantic web applications to package and format content.

The authoring process will, he says, need to include creating taxonomies to help a computer understand the content, its context, and appropriate formats for display of it, and store this in a schema. It then needs to have the ability to create processing rules that dictate how to process content of a specific type into a defined format.

7.17. Authoring performance support applications

Performance support application development is surging, especially for the mobile platform, due the "always on, always with you" nature. Although the instructional design process and end-use is different for performance support vs eLearning, there are no reasons why standard eLearning tools and techniques cannot be used to develop performance support. However, some of the differences between performance support tools and eLearning can drive emphasis of the following features when choosing an authoring tool:

- Robust search capability. This can include content stored locally in the application as well as repositories of web-based content.
- Lack of need for assessments, with the possible exception of self-assessments.
- Workflow, checklist, and timeline-based screen templates may be needed. Decision support trees may also be helpful.
- The ability to use the tool in either standalone disconnected or connected mode, if users will be in field environments where connectivity is absent.
- The ability to send usage data to a web service to enable stakeholders to easily evaluate tool effectiveness and diagnose performance gaps (where performance support tool emphasis may be needed).

If your organization is involved in building performance support applications, you may want to limit your choice of authoring tools to those that robustly support these features.

Currently, there are no authoring tools advertised or designed specifically for authoring performance support, although this may change due to the popularity of performance support in the workplace.

7.18. HTML5 format

The Adobe Flash® format has dominated the interactive multimedia and eLearning landscape since 1996. It has been used to create countless media-rich, interactive Level 3 and Level 4 eLearning courses, as well as animations and videos appearing as media assets within a variety of learning objects. Many authoring tools output to Flash format simply because of its near unlimited ability, via its ActionScript scripting language, to handle extensive interactivity in Rich Internet Applications (RIAs). It has also been the format of choice for Internet videos (via .flv format).

Flash has recently experienced a downturn in popularity and support, however, in favor of what is known as HTML5 (the combination of CSS3, HTML v.5, and JavaScript) for a variety of reasons:

- Apple has never supported Flash on its iOS mobile devices, and these devices (especially iPads, where rich media content is less constrained by a smaller screen) have become vastly more popular (including for mLearning).
- Performance and instability issues with Flash.
- Security issues with Flash that inherently limit the ability of a plug-in application-based web object (like Flash) to control and communicate with web pages and the browser (especially its parent web page).
- Usability issues with Flash in a browser context that, for instance, renders use of the Back and Forward buttons confusing.
- The steep learning curve to learn Adobe Flash authoring. Many programs simpler than Flash are available to create Flash objects, but to fully take advantage of its features, it is necessary to learn the Flash program.

Adobe started going down a path of deprecating Flash in 2011, culminating with their withdrawal of support for Flash Mobile (for Android devices) in June 2012.

HTML5 is now widely touted (and seemingly accepted by Adobe) as the replacement for Flash due to the fact that it is designed as the new native web authoring language. It is not a fully completed specification—it will probably remain in progress for a number of years—but browsers have nevertheless adopted many parts of the draft spec already.

There are fundamental advantages to using a native web language (HTML) vs a plug-in application language, as follows:

- HTML content can more easily be made accessible to screen readers.
- There is no plug-in application that needs to be continuously updated. This can be a problem in managed IT environments where new versions must go through lengthy approval processes and users must rely on IT staff to upgrade their system.
- HTML content is far easier to edit. HTML only requires a text editor. In Flash, editing requires making changes to the source files in the source application. The output files (.swf) and the editing files (.fla) are different formats, and you cannot edit the output files in the Flash software. In HTML, there does not necessarily need to be a different source file format from the output file format; if you use a "round trip" WYSIWYG web page editor such as Dreamweaver®, you can reimport outputted files into the web page editor and edit them at any time.
- HTML is generally easier to hand code than a plug-in language like ActionScript (though this depends on how much JavaScript is used), reducing development costs.

- Security issues are lessened because the browser does not see HTML code as coming from a "foreign" application.
- It is difficult to configure plug-in content to be searchable by external search engines, whereas HTML code is always searchable by default.
- It is easier to translate native HTML code, or at least expose the contents of the web page to translation engines.
- In general, it is harder to create a seamless user experience when users navigate from the browser environment to the plug-in environment; the plug-in environment tends to be more functionally self-contained (it needs to be since the code base is different).

Furthermore, there are particular advantages offered by HTML5 (vs earlier versions of HTML):

- Audio and video can be streamed natively in HTML5.
- Programmers have new structural elements in HTML5 that allow code to be more efficiently organized, as well as features that improve interoperability.
- Validation of user input in forms is a built-in feature.
- Bandwidth-efficient vector graphics (via SVG format) are natively supported.
- HTML5 allows local data storage, which can be accessed to support the web application.
- Drag and drop interactions are natively supported.
- Geolocation features of a mobile device can be leveraged.

Perhaps the biggest advantage of HTML5 to eLearning is that it allows "responsive design" for mobile devices, meaning that the content is dynamically resized based on the size of the browser window. Add to this the fact that it is supported by iOS, and it is clearly the best strategy for delivering eLearning to many mobile audiences.

Should you look for an authoring tool that outputs HTML5? It depends on your audience. If you are delivering eLearning to users outside of your organization (i.e., you cannot easily baseline the browsers that will be used), you may want to err on the conservative side and skip it for now, since users may not have a browser that can handle it (or certain parts of the spec at least). But the day is fast approaching when browsers will support it fully in its current draft state.

For further information on the impact of HTML5 on eLearning and how to make the decision as to whether to adopt it as your eLearning format (and consequently choose an authoring tool to support it), see the Elearning Guild report on HTML5 at http://www.elearningguild.com/content.cfm?selection=doc.2574.

As of this writing there are very few authoring tools that support output to HTML5. The ones that do may not offer all of the features and advantages of HTML5 that are implemented by browsers. Check with a vendor to specifically identify which features of HTML5 are supported and which are not before you purchase a tool that advertises the ability to output in HTML5 format. The following is a preliminary list of authoring tools that support HTML5 from Ganci (2013):

- Adobe Captivate 7
- Articulate Storyline 1, Update 3
- Composica Enterprise 6
- Claro
- iSpring Suite

- Landmark Liquid
- Lectora Inspire 11.1
- ReadyGo
- SmartBuilder

7.19. Interactive video

Interactive video is quickly becoming a mainstream type of content object for learning applications. It leverages the popularity of video as an effective medium for learning by adding interactivity to the video.

The idea of associating menus, links, and (semitransparent or opaque) buttons/hotspots with a video has always been possible within web pages that launch videos. However, in interactive video, the navigation controls are superimposed over areas of the video itself while it is running, appearing at strategic points, rather than placed around the video on the web page. The navigation controls can navigate to other related video clips (including other interactive videos), other types of content (such as PDFs), or web sites. Forms, text entry fields, and assessments can also be overlaid on the video.

Westfall (2015) reports that use of interactive video can stir learners to take action as a result of richer, more visual media or interactive videos, increase participation rates, and create a more engaged audience.

The following are examples of learning applications of interactive video:

- Assessment questions asked at strategic points in the video
- Links to additional resources
- Chapter overlays for learner navigation to topics of interest within the video
- Invisible hot spots over areas of the video to pop up further information about that object or area
- Multiple timelines presented for branching scenarios
- User annotations
- Video storytelling, with learners having the ability to choose alternative paths through the story

Interactive video provides a seamless, immersive experience for the user and makes “choose your own adventure” adaptive learning scenarios, as well as “drill down” options for more detailed or relevant information, more user friendly. It also makes tracking the user’s interactions with the video easier.

Interactive video usually connotes use on mobile devices, since that is the main platform for videos for learning currently. One especially compelling implementation for mobile is to superimpose semi-transparent hot spots that allow such things as scrubbing quickly forwards or backwards through the video, as a convenience for “fat finger” mobile phone navigation (via thumbs, etc.), rather than traditional small buttons in the interface.

Building interactive videos requires the videos to be played within a system, like an LMS, that inserts the superimposed elements while the video is playing, or access to custom controls (usually HTML-based) of the video player software. In the latter case, options may be limited to controlling parameters of playback, but not actually superimposing images on the video; links and interactive widgets may need to appear around (i.e., in the player interface), but not over, the video.

Simple interactive videos can be created within YouTube, using their built-in editing tools. Links can be superimposed to link to the next video in a series, for instance. However, if you want advanced features like those described above, they need to be created within an authoring tool.

7.20. Social video

Social video is a way to effectively crowd source the addition of content relating to a particular video. Learner comments on video often fill important content gaps.

Social video involves adding a scrolling bar on the side of the video that is synched with the video such that user comments are tagged to a particular point in the video transcript, which is associated with a location in the video. When a user clicks a comment, the video and associated transcript skip to the location where the comment was tagged. Comments scroll as the video plays in order to synch the comments with the position in the video and transcript they are tagged to and comments can be bookmarked.

Social video relies on producing a transcript of the video (not necessarily the same as closed captioning, which is often not a separate text file but embedded within the video file). Producing video transcripts is not hard nowadays; speech to text converters are reasonably accurate and automate this process at relatively low cost. Clicking comment indicators in the transcript also navigates the user to an associated comment.

7.21. Microlearning video

Microlearning videos are a subset and the most common implementation of microlearning. Microlearning is a concept that largely emerged with the advent of mLearning. It stems from the fact that short, self-contained pieces of content are better suited for the mobile platform, as opposed to entire courses. The term “microlearning” has particular connotations within the eLearning industry, but “microlearning” can be broadly applied to any learning asset of about 5 minutes or less duration. It is often associated with blended learning, where mixed-mode microlearning assets can combined flexibly within a learning experience. These “blended” assets could be a mix of performance support and training modules and well as eLearning and instructor-led training. Short videos often form the backbone of solutions involving microlearning. Microlearning videos can be of various lengths, depending on the limitations of the platform. Here are some examples:

- YouTube® = 10 minutes
- Twitter® = 30 seconds
- Instragram® = 15 seconds
- Vine® = 6 seconds

It generally does not make sense for videos of 15 seconds or less to have an audio track. To convey the learning message effectively, this makes it even more important for them to be carefully scripted.

Ultra short length microlearning videos often show a process in fast motion, with the ability to click to step through it in normal or slow motion. It is also very important to tag microlearning videos, or microlearning of any kind, with metadata so that individual microlearning content pieces can be assembled into a meaningful whole learning experience.

Microlearning videos can be created by training stakeholders or users, empowering users to create tutorials on subjects within their expertise and share them with others.

Authoring tools and LCMSs (called “video content management systems” - VCMSs), for instance, KZO®, are now appearing that are optimized specifically to create microlearning-optimized videos, incorporating all of the considerations mentioned.

If you are considering creating microlearning videos, it is very important that you consider the content management aspect, in order to deal with issues such as:

- The difficulty of capturing usage tracking for downloaded videos. A VCMS, by the fact that videos are streamed from it, is positioned to handle detailed usage tracking.
- Rules and permissions for creation and use of videos, especially user-generated videos. Some regulatory environments (like medical, with HIPAA requirements) may have strict rules for details that can be used in content, or need to know based permissions for viewing them.
- Pushing videos to users (perhaps in a "daily drip") rather than simply offering them in a "pull"-based library

For more information on content management systems for microlearning, see ADL's *Choosing an LMS* white paper (Berking , 2015) available at:
http://adlnet.gov/adl-assets/uploads/2016/01/ChoosingAnLMS.docx

7.22. Crowd sourced authoring systems

Crowd-sourced authoring systems are emerging (for example, Oppia®) that gather and compile data on how learners interact with it, making it easy for authors to spot and fix shortcomings in a lesson. These systems identify responses that learners are giving to questions that the system is not responding to adequately, allowing authors to create a new learning path for it based on what they would actually say if they were interacting in-person with the learner. This allows the system to accumulate the collective wisdom of course authors and incrementally improve the learning.

Systems have been available for some time now in which interaction widgets can be uploaded and made available to the community of authors using that tool (for example, ZebraZapps®). The crowd sourcing referred to here is different in that it deals with the pedagogical aspect of the content, rather than the technical mechanics of rendering it.

7.23. Intelligent content

Learning professionals of many stripes, especially learning content and technology providers, are converging on the idea of "intelligent content" as a way to increase flexibility and efficiency in creating and managing learning experiences. ADL has been promoting the intelligent content paradigm for some time, with the creation of SCORM and its predication on interoperable, reusable content.

Intelligent content is defined in Rockley (2015) as "…modular, structured, reusable, format-free, and semantically rich, and, as a consequence, discoverable, reconfigurable, and adaptable."(p.1). Intelligent content is achieved by doing some or all of the following:

- Tagging with semantically rich metadata
- Modularizing into discrete components using a consistent scheme and uniform structure
- Separating raw information from formatting/presentation instructions

One important result that is now fairly well established is the ability to mix and match content components, such as in learning content management systems (LCMSs). In LCMSs, content objects are loaded in the LCMSs content repository, or library, and learning products are assembled dynamically at runtime based on the stated needs of the user and their demographic profile (see Berking, 2015 and 3.2 *Learning content management systems (LCMSs)*). For instance, the user can request a Participant Guide for a classroom course or an eLearning module, assembled from the same objects that a content author has only had to author once. It is also extremely important, in this day and age of mobile devices, to make content consumable across multiple formats (paper, web, apps), devices (tablets, smartphones, desktops), and systems (LMSs, LCMSs, content brokering systems, intelligent tutoring systems).

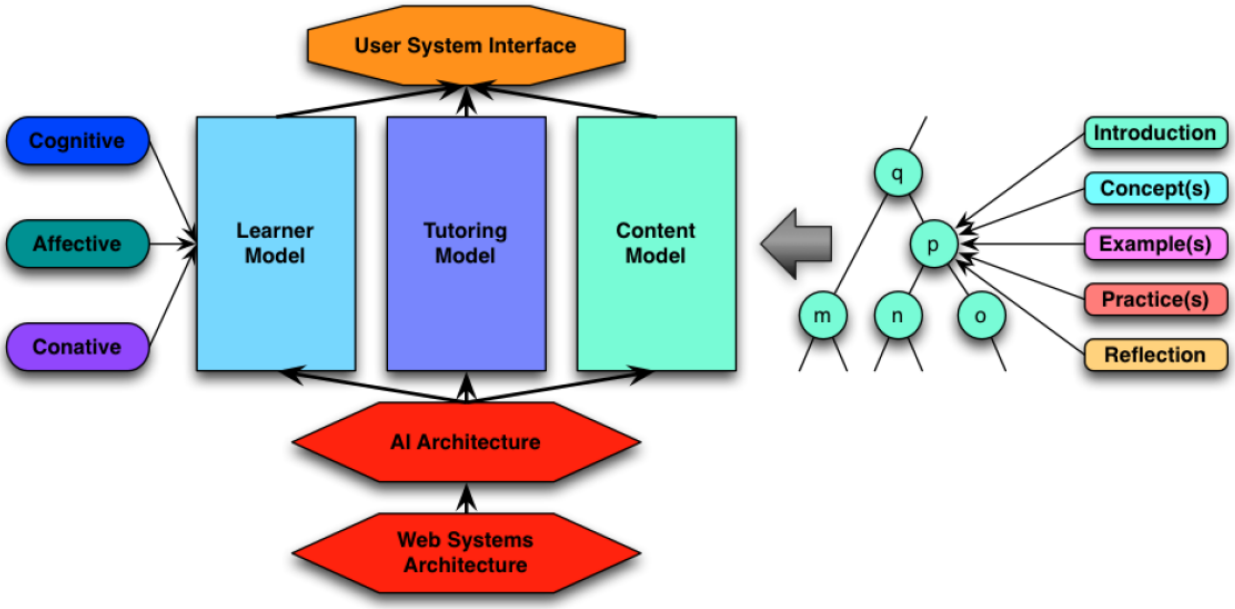
Authoring tools are currently open-ended enough in their architecture to fully support the intelligent content concept through their ability to tag content with rich metadata, modularize using templates, and

use CSS to separate content from appearance. However, it remains to be seen if future interoperability standards for intelligent content will change requirements for authoring tools to support intelligent content. These standards may end up being consolidated into a single overarching standard or reference model for intelligent content.

A starting point for an intelligent content architecture is described by Quinn (2015). He advocates creating and tagging content objects according to the following core elements of a learning experience:

- Introduction
- Concept(s)
- Example(s)
- Practice
- Reflection

These could be manipulated within a content brokering or adaptive learning system as follows:



From Quinn, 2015, p.17.

This example paradigm underlines an important point about authoring tools; due to the drive towards standardizing not only content objects but the overarching learning architecture or environment itself, authoring tools will need to be designed at least to leverage synergies of content design with the entire environment; in fact, it is likely that authoring tools will need to be designed explicitly to work within such an environment, with carefully prescribed inputs, outputs, and goals.

8. Process for choosing tools

ADL recommends the following high-level process for choosing authoring tools. This process should be first applied to the primary tool you will use for authoring, then separately for each secondary or auxiliary tool. Once you have gone through the requirements definition exercise in the process below and selected a primary tool, you should then know what gaps you need to fill by acquiring secondary tools (for example, for asset production):

1. Determine your high-level requirements. It is important to stick to only the critical, high-level, and highly differentiating requirements at this point. That will serve to quickly filter many unsuitable candidates when you get to Step 4 below. This may require a formal requirements

definition effort, especially if you are a large enterprise with many different organizations who may have different (and hard to predict) needs from your organization.

Be aware that there are many types of requirements (functional, usability, etc.), representing different points of view (users, administrators, stakeholders, etc.). See Wiegers's (2000) article (available at http://processimpact.com/articles/reqtraps.html) for information on how to avoid "requirements traps" such as ambiguous or vague definitions.

2. Your high-level requirements should focus on the following areas:

   - Target platform
     - Desktop computer
     - Tablet
     - Mobile phone
   - Type(s) of training (sometimes multiple types are required in your organization)
     - Asynchronous eLearning
     - Synchronous virtual classroom or virtual world
     - Asynchronous virtual classroom (for example, recorded synchronous classroom sessions)
     - Instructor-led training (ILT) with certain aspects delivered electronically (for example, assessments)
   - Particular learning functions needed, especially social learning functions such as wikis, blogs, forums, and chat.
   - Media
     - Audio
     - Video
     - Graphics
     - 2D animation
     - 3D animation
   - Level of interactivity
     1. Passive—no interactivity except to navigate to next screen
     2. Simple interactions limited to elaboration of information or getting feedback
     3. Adaptive navigation and branching
     4. Highly interactive simulation with granular assessment and adaptive learning paths
   - Skill sets of authors. Generally, your authors will fall into these groups:
     - Instructional designers
     - SMEs
     - Junior developers
     - Senior developers

     Skill sets should be matched to the power and complexity of the tool you choose. For instance, you would not want to give an easy-to-learn but simplistic, limited-functionality tool to senior developers, since they would be hamstrung and frustrated using the tool.
   - Need for non-technical staff to edit content (this is especially important where content changes frequently or client wants to take over content maintenance responsibilities)
   - Output file format (see 4.7. *File formats*)
   - Standards compliance for output files (see 4.10. *Standards support*)
   - Kinds and levels of support and training required by the tool
   - Interworking and/or compatibility with other tools or software you will be using

- Collaborative authoring (vs standalone authoring)
- Number, roles, and distribution of potential tool users
- Bandwidth and other IT constraints and opportunities

3. Determine your budget for purchasing the tool and associated support/training contracts. This includes any customization, special features, or adjustments to your IT environment that you predict you will need.
4. Determine categories of tools you will need (see 3. *Categories and examples of authoring tools*). Because these categories overlap, you may identify more than one category for consideration.
5. Identify specific tools for the key categories identified in the previous step (see 3. *Categories and examples of authoring tools* for example tools in each category). You may decide at this point to develop your own product rather than purchase a commercial off-the-shelf (COTS) product or acquire an open source product. Note that if you are a U.S. government entity, the government acquisition process requires justifications for acquisition choices. You will need to validate or justify your decision to develop your own tool.
6. Develop and complete a matrix that allows assessing the tools identified in Step 5 against your requirements developed in Step 1. See *Appendix A: Sample Tool Requirements Matrix* for a sample. You may want to complete a separate matrix for each different category of tools you have identified as a requirement for your organization, since each category of tools has its own distinct parameters and typical feature sets. You may need to acquire different toolsets for different types of projects in your organization.
7. Filter the list of potential candidates, eliminating those that do not meet your minimum requirements and/or are over your budget. Create and send Requests for Proposals (RFPs) to the final candidates at this point, if that is required by your acquisition process.
8. Compile a detailed and complete features list for all of the remaining candidate tools. You may want to develop this list from sampling one tool that seems to be the most feature-rich, and add any features uncovered by your analysis of other systems as you complete the comparison process. Or, you can use part or all of the criteria mentioned in 5. *List of possible requirements for authoring tools* as your features list. You may want to edit this list of features to only those that you care about now; however, this may be limiting since you may be unfamiliar with the usefulness of some features, or they may become useful in the future.
9. Develop a matrix (see the *Appendix B: Sample Tool Features Rating Matrix* for a sample) that compares the systems identified in Step 7 using the features list developed in Step 8. Complete as much of this matrix as possible from the tools' documentation; if you need more information, ask their sales representatives for it. Assign a numerical rating for each cell in the matrix, indicating the degree of implementation of that feature (which could be 0 if it does not have that feature). The matrix should weigh each feature according to its importance to you, enabling a rollup score for each tool.
10. Contact the top scoring vendors (three to five is a reasonable number) from the previous step and ask for a live presentation/demo. Ask the vendor for a demonstration in your facility, running their system in your IT environment. The vendor may want to present a canned demo of their product using a presentation format like PowerPoint® or Flash®, and that is fine as a general overview of the tool's capabilities, but you should see how well the system expresses these capabilities within your IT environment and with your content (if you need to be able to edit legacy content in the tool).
11. Make your decision based on the results of the previous step, taking into account the total cost of ownership (TCO), including the application, training, upgrades, maintenance, and any intangible

items. Consider whether a hosted (see 4.4. *Hosted solutions*) solution is right for you, if you are considering web-based tools and if a hosted solution is available from the vendor.

9. For more information about authoring tools

- Bersin & Associates
  www.bersin.com
  This company offers a variety of reports on aspects of eLearning, including authoring tools.
- Brandon Hall Group
  http://www.brandon-hall.com
  This company sells research reports containing trends and profiles of authoring tool products, a selection utility, and a comparison utility.
- Centre for Learning and Performance Technologies. Directory of Learning Tools.
  http://c4lpt.co.uk/directory-of-learning-performance-tools/instructional-tools/.
  This web site contains a detailed list of available authoring tools, with abstracts describing each.
- ELearning Centre (UK).
  http://www.e-learningcentre.co.uk/eclipse/vendors/authoring.htm.
  Web site that contains a detailed list of available authoring tools with abstracts describing each.
- ELearning Guild
  http://www.elearningguild.com

  This trade association offers buyer's guides and trend reports on authoring tools and other aspects of eLearning.
- Fenrich, P. (2005). Creating Instructional Multimedia Solutions: Practical Guidelines for the Real World. Santa Rosa, CA: Informing Science Institute.
  This book contains a chapter about comparing, contrasting, and evaluating authoring tools.
- TRADOC Capability Manager for the Army Distributed Learning Program (TCM-TADLP)
  http://www.atsc.army.mil/tadlp/index.asp.
  This web site contains comprehensive information for anyone involved in designing and developing technology-based training for the U.S. Army.
- Training & Education Developer Toolbox (TED-T)
  https://atn.army.mil/TreeViewCStab.aspx?loadTierID=2904&docID=35.
  This site is not an authoring tool itself, but has helpful technical information for U.S. DoD developers. It is available to U.S. DoD users only. It requires a Common Access Card (CAC) to log in since it is on the Army Training Network (ATN).
- Trainer's Guide to Authoring Tools (Training Media Review)
  http://www.tmreview.com/ResearchReports/ .
  Contains ratings of tools.

10. References cited in this paper

- Allen, M. (2012). Michael Allen's ELearning Annual 2012. San Francisco: Pfeiffer Publishing.
- Berking, P. (2015). Choosing an LMS. ADL white paper available at
  http://adlnet.gov/adl-assets/uploads/2016/01/ChoosingAnLMS.docx

- Elearning Guild. (2015). Authoring Tools for Mobile Design. Research article. Retrieved 6/24/15 from http://www.elearningguild.com/content.cfm?selection=doc.3971
- Haag, J. (2011). ADL Mobile Learning Workshop 29 Aug 2011. (presentation slides)
- Instructional Design Guru. (2011). Mobile app (for iPhone)
- Lee, C.S. (2014). Why Responsive Design? *Elearning Magazine* July/August 2014. 6(2), 40. Retrieved 12/17/14 from http://gelmezine.epubxp.com/i/350882/40.
- Quinn (2011). Designing mLearning. San Francisco: Pfeiffer Publishing.
- Quinn, C. (2015). *It's Time to Do Learning Like Grown-ups: Content Systems.* DevLearn 2015 conference presentation. Retrieved 11/20/15 from http://www.elearningguild.com/conference-archive/index.cfm?id=6941&from=content&mode=filter&source=sessions&showpage=6&sort=titleasc&type=DevLearn+2015+Conference+%26+Expo
- Rockley, A., Cooper, C., and Abel, S. (2015). Intelligent Content: A Primer. Laguna Hills, CA: XML Press.
- Shank, P., & Ganci, J. (2013). eLearning Authoring Tools 2013: What We're Using, What We Want (eLearning Guild Survey Report 2013). Available at http://www.elearningguild.com/research/archives/index.cfm?id=170&action=viewonly&from=home
- Tozman, R. (2012). Why ELearning Must Change: A Call to End Rapid Development. In *M. Allen (Ed.) Michael Allen's ELearning Annual 2012*. San Francisco: Pfeiffer Publishing.
- TrainingIndustry.com (2015). Web taxonomy retrieved 6/25/15 from http://www.trainingindustry.com/taxonomy.aspx
- Udell (2012). Learning Everywhere. Nashville: Rockbench Publishing.
- Weigers (2000). Karl Wiegers Describes 10 Requirements Traps to Avoid. Web article retrieved 12/16/14 from http://processimpact.com/articles/reqtraps.html
- Westfall, David, interviewed by Christopher P. Skroupa. Leveraging Talent: Mindset Over Skillset. Forbes. 11 May 2015. Retrieved 7/6/16 from http://www.forbes.com/sites/christopherskroupa/2015/05/11/leveraging-talent-mindset-over-skillset/#64c0b06960f2

Appendix

A. Sample Tool Requirements Matrix

The following is a sample of a matrix that can be used in step 6 presented in 8. *Process for choosing tools.* The step is described as:

Develop and complete a matrix that allows assessing the tools identified in step 5 against your requirements developed in step 1.

To use the matrix:

1. Enter items you have determined to be your high-level requirements for the tools as row labels in the "High-level requirements" column.
2. Enter the product names at the top of each column, replacing "LMS product 1", "LMS product 2", etc..
3. Research and complete the cells with information indicating whether each product meets that requirement (may be "yes" or "no", a more lengthy description of how it meets or doesn't meet the requirement, or a number that roughly quantifies the degree to which that requirement is supported in the product).

Tool Requirements Matrix								
	Tool product 1	**Tool product 2**	**Tool product 3**	**Tool product 4**	**Tool product 5**	**Tool product 6**	**Tool product 7**	**Tool product 8**
High-level Requirements								

B. Sample Tool Features Rating Matrix

The following is a sample of a matrix that can be used in step 8 presented in in 8. *Process for choosing tools.* The step is described as:

Develop a matrix that compares the systems identified in step 7 using the features list developed in step 8. Complete as much of this matrix as possible from the tools' documentation; if you need more information, ask their sales representatives for it. Assign a numerical rating for each cell in the matrix, indicating degree of implementation of that feature (which could be 0 if it does not have that feature). The matrix should weight each feature according to its importance to you, enabling a rollup score for each tool.

To use the matrix:

1. Replace the top row (Tool product 1, Tool product 2, etc.) with the names of the systems you have identified for consideration.
2. Replace the row names (Feature 1, Feature 2, etc.) with the names of features you have identified as requirements.
3. For each Weighting factor cell in the column to the right of the Feature name, replace the text with a number between 1-3 to weight the relative importance of that feature to your organization (the higher the number, the more important).
4. Research the feature information for each system and complete the cells with the number indicating the degree to which each system has that feature. We suggest 0-2, 0 being "does not have that feature" and 2 being "has implemented this feature to the fullest extent possible". You may want to use a rubric developed by Brandon-Hall (Brandon-Hall/Saba webinar "Selecting an LMS" 9/14/10) that rates the feature in terms of how "out of the box" it is. Assigning numbers to their rubric would yield the following rating scale:
    a. 5=Automatic (built-in, out of the box feature)
    b. 4=Semi-automatic (mostly built-in, but requires some programming or customization to activate)
    c. 3=Semi-custom (partially available. The system can be adapted to implement this feature through moderate customization)
    d. 2=Custom (not available but can be added, possibly at high cost, with programming)
    e. 1=Not available (would be impossible or cost-prohibitive to customize the system to add the feature due to incompatibilities with system architecture, etc.)

    If a feature is not available, you may also want to note in this matrix whether a feature is available from another vendor as an add-on, so as not to totally rule out/penalize the vendor for lack of that feature. This can be incorporated into the rating scale such that a rating of "3" means that a feature is available as a third party add-on.
5. The rollup score row at the bottom will provide the total weighted score for each system (right-click on it and select **Update Field** after you make any changes to the weighting values or ratings).
6. If you add columns or rows, copy and paste the Rollup score formula and adjust the row and column references in the formula accordingly. Right-click the pasted Rollup score and select **Toggle Field Codes** to see and edit the formula.

Authoring Tool Features Rating Matrix						
Feature name	*Weighting factor*	**Tool product 1**	**Tool product 2**	**Tool product 3**	**Tool product 4**	**Tool product 5**
Feature 1						
Feature 2						
Feature 3						
Feature 4						
Feature 5						
Feature 6						
Feature 7						
Feature 8						
Feature 9						
Feature 10						
	Rollup score	0	0	0	0	0