WBT – Stand und Perspektiven

Prof. Dr. M. Mühlhäuser, Technische Universität Darmstadt

Einführung

Web-Based Training (WBT) wird häufig als Allheilmittel angesehen, wenn es darum geht, den ständig wachsenden Bedarf der Informationsgesellschaft nach lebenslanger Aus- und Weiterbildung zu befriedigen. Die Erstellung von WBT-Material (nachfolgend auch „Kurse“ genannt) vereinigt Aspekte von Autorentätigkeiten und der Software-Entwicklung mit Aspekten von Pädagogik und Didaktik.

Ein Blick auf den Stand Technik zeigt, dass sich WBT heute nur wenig vom seit Jahren bekannten CBT (Computer-Based Training) unterscheidet: 1. die Verbreitung findet über das Internet statt über CD-ROM statt, 2. die verfügbare Web-Technologie wird verwendet, allerdings bislang noch mit wenig Nutzen für den Endanwender (siehe unten). Dieser Beitrag besteht aus zwei Teilen. Teil 1 analysiert den Stand heutiger WBT-Autorenwerkzeuge, wobei wesentliche Anforderungen herausgearbeitet und zum Vergleich gängiger Konzepte und Systeme verwendet werden. Teil 2 versucht, über den heutigen Stand hinauszugehen und mögliche fortgeschrittene Formen des WBT aufzuzeigen. Es erweist sich dabei als schwierig, solche fortgeschrittenen Anwendungen mit der heutigen Web-Technologie zu realisieren. Daher wird am Schluss kurz darauf eingegangen, wie die kommende Generation von Web-Technologie nutzbringend eingesetzt werden kann.

Nachfolgend gehen wir bei WBT-Material von drei Ebenen aus, vgl. Abb. 1:

1. Multimediale Inhalte als unterste Schicht; sie repräsentieren die atomaren Einheiten des Domänenwissens
2. Instruktionelle Transaktionen (ITA) als mittlere Schicht; diese stellen die zentralen Bausteine dar, welche WBT-Autorenwerkzeuge handhaben
3. Instruktionelle Strategien (z.B. für WBT-Typen wie „Tutorium“ oder Theorien wie „progressive deepening“); diese nennen wir auch „Makro-Strategien“.

Abb. 1 macht deutlich, dass der Einsatz von WBT-Autorenwerkzeugen nur der letzte Schritt im Entwicklungszyklus von WBT-Material ist. Ihm voraus gehen Schritte wie die folgenden:

- didaktische Planung mit Festlegung der Lernziel-Hierarchie (dazu gehört z.B. die Zielgruppenanalyse) und der einzusetzenden Makrostrategien
- Entwurf und Inhaltserschließung inkl. Semantik (z.B. mittels Konzeptgraphen) und „Syntax“ (Erstellung bzw. Aufnahme der Medien d.h. Inhalte)
- detaillierte Kursplanung einschließlich Festlegung der Kursmodule, ihrer Beziehungen zu Makrostrategien, ihrer Einbettung in ITAs etc.

Gängige WBT-Autorenwerkzeuge unterstützen meist die frühen Entwicklungsphasen nicht direkt. Sie werden allerdings zunehmend in Entwicklungsplattformen eingebettet, die eine Verbesserung bringen, oft aber keinen befriedigenden Integrationsgrad.

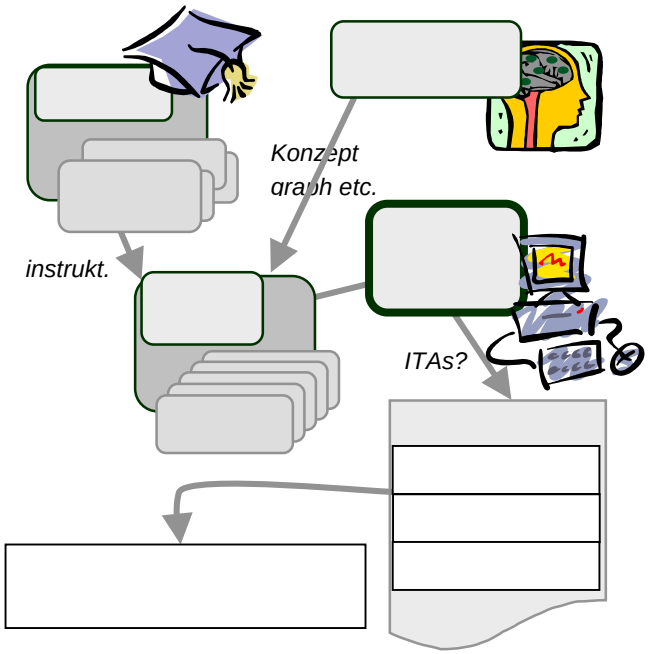

Abb. 1: WBT-Lebenszyklus, vordere Phasen (‚upper CASE’)

2. WBT-Stand und Autorenwerkzeuge

Nachfolgend werden wir heutige WBT-Werkzeuge nach folgenden Kriterien unterscheiden: *verwendete Metapher*, *ITA-Unterstützungsgrad* und *allgemeine Klasse*.

Verwendete Metapher: eine wichtige Unterscheidung betrifft die grundlegende verwendete Metapher zur Organisation des Kurses und zur Anordnung der Vielzahl verwendeter ITAs. Für unsere Zwecke reicht es aus, drei Klassen zu unterscheiden:

- *Buch/Karten-Metapher:* viele Werkzeuge organisieren Kurse als „Bücher, bestehend aus Seiten“ oder „Stapel von Karten“, wobei die Seiten bzw. Karten bei der Kursausführung einzelnen Bildschirmseiten entsprechen. Die sequenzielle Organisation des Kurses als Buch bzw. Stapel bestimmt die bevorzugte Ablaufreihenfolge, Abweichungen von dieser Reihenfolge müssen meist umständlich festgelegt werden. Offensichtlich haben die Metaphern „Buch“ und „Kartenstapel“ viel gemeinsam, weshalb sie hier zusammengefasst werden.
- *Zeitachsen-Metapher:* hierbei wird der Kursablauf als Handlungsstrang interpretiert, wie z.B. ein Kinofilm. Auch hier dominiert also eine bestimmte sequenzielle Ablaufreihenfolge. Während bei der Buch/Karten-Metapher kontinuierliche Medien (wie Video) normalerweise als Teil einer Seite ablaufen und die Folgeseite explizit aufgerufen werden muss (z.B. durch Drücken eines Buttons), sind es bei der Zeitachsen-Metapher eher die Unterbrechungen im kontinuierlichen Ablauf, die vom Entwickler extra vorgesehen werden müssen.
- *Icon-Fluß-Metapher:* hier kann und muß der Autor den Kursablauf explizit festlegen, indem er atomare Kurselemente (die i.a. genau den ITAs entsprechen) als Kontrollfluß anordnet. Berechnungen im Hintergrund, die z.B. das Benutzermodell verwalten, können relativ elegant mit dem Kontrollfluß verknüpft werden. Für fortgeschrittene instruktionelle Strategien und erfahrene Entwickler eröffnet diese Metapher offensichtlich die besten Möglichkeiten. Die anderen beiden Metaphern verschaffen solchen Autoren einen leichten Einstieg, die vorher vor allem mit Printmedien wie Lehrbüchern (Buch/Karten-Metapher) bzw. mit kontinuierlichen Medien wie Lehrfilmen (Zeitachsen-Metapher) zu tun hatten.

ITA-Unterstützung: Instruktionelle Transaktionen als Kern der Ebene 2 können grob in drei Klassen eingeteilt (und Werkzeuge nach deren Unterstützung bewertet) werden:

Präsentations-ITAs bilden die offensichtlichste Klasse; sie dienen oft 1:1 der Einbindung der Medien aus Ebene 1 (s.o.). WBT-Werkzeuge unterscheiden sich in der Unterstützung von Präsentations-ITAs vor allem hinsichtlich der unterstützten Medien und –formate. Während eine zu geringe Zahl unterstützter Medienformate leicht durch externe Werkzeuge kompensiert werden kann, sind Mängel hinsichtlich Medien*typen* meist sehr viel hinderlicher: wenn z.B. Animationen nur unzureichend unterstützt werden, kann dies die Qualität von WBT-Material deutlich verringern.

Interaktions-ITAs sind quasi die Würze eines WBT-Werkzeugs, da ihre Verwendung wesentlich den Grad bestimmt, zu dem Lerner aktiv eingebunden werden – was wiederum maßgeblichen Einfluß auf Motivation, Aufmerksamkeit und Retentionsrate hat. Wieder gibt es erhebliche Unterschiede, fortgeschrittene Werkzeuge unterstüten z.B. natürlichsprachliche Antwort-Texte, „drag-and-drop"-Platzierungsaufgaben u.v.a.

Kontrollfluß-ITAs entsprechen den wesentlichen Kontrollfluß-Konstrukten von Programmiersprachen, wie Verzweigungen und Fallunterscheidungen. Bei Werkzeugen nach der Icon-Fluß-Metapher sind sie 1:1 in Icons umgesetzt, bei den anderen Metaphern oftmals bis auf wenige Ausnahmen nur über Skriptsprachen (und damit nur für erfahrene Autoren) zugänglich.

Mangelnde Unterstützung instruktioneller Strategien: in der vorgenannten Diskussion fehlt fast vollständig die oben erwähnte dritte Ebene, die der Makro-Strategien. In der Tat ist entsprechende explizite Unterstützung in gängigen WBT-Werkzeugen kaum zu finden. Stattdessen wird davon ausgegangen, dass Autoren solche Strategien außerhalb des Werkzeugs (ggf. auf Papier) planen und während der Realisierung quasi im Kopf haben. Dieser Mangel ist nach Meinung des Autors hauptverantwortlich dafür, dass sich bislang instruktionelle Strategien für computergestützte Aus- und Weiterbildung nur relativ langsam entwickelt haben und dass anspruchsvolle Strategien mit wenigen Ausnahmen kaum Eingang in den Bereich WBT gefunden haben (s.u.).

Allgemeine Klasse: da WBT-Material naturgemäß für das Web entwickelt wird, kann zur Kursentwicklung im Prinzip ein allgemeines Web-Entwicklungswerkzeug verwendet werden. Daneben gibt es zunehmend spezialisierte WBT-Autorenwerkzeuge. Am meisten wird allerdings bis heute eine dritte Klasse verwendet: die der CBT-Autorenwerkzeuge, welche um das Web als „Vertriebskanal" erweitert wurden. Die drei genannten Klassen sollen noch etwas näher beleuchtet werden. Soweit nachfolgend beispielhaft marktgängige Werkzeuge zitiert werden, ist zweierlei zu beachten:

einerseits schreitet die Entwicklung laufend fort, so dass aufgezählte Einschränkungen zum Zeitpunkt der Verbreitung dieses Artikels ggf. überholt sind; andererseits dienen die Nennungen vor allem der besseren Orientierung der Leser und sind keinesfalls als Testergebnisse oder gar Kaufempfehlungen zu werten –detaillierte vergleichende Evaluationen wurden zwar von der Gruppe des Autors durchgeführt, sind aber nicht Gegenstand dieser Veröffentlichung.

Allgemeine Web-Entwicklungswerkzeuge: nimmt man die Sichtweise ein, dass ein HTML-Dokument eine Seite oder Karte (quasi ohne Größenbeschränkung) darstellt, dann kann man Standard-Web-Entwicklungswerkzeuge in die Kategorie der Werkzeuge nach Buch/Karten-Metapher einordnen. Alle pädagogisch-didaktischen oder sonst auf Lernsoftware bezogenen Aspekte werden naturgemäß in solchen Werkzeugen vernachlässigt. Dies betrifft z.B. spezielle ITAs für lern-zentrierte Medien und betraf bis vor kurzem auch Interaktions-ITAs generell. Der zunehmend interaktive Charakter des Web hat diesen Aspekt verbessert, während einige Nachteile dieser Werkzeug-Klasse (für die Kursentwicklung) wohl bestehen bleiben werden, z.B. so gut wie keine Integration mit (auf Lernsoftware bezogenen) Werkzeugen aus vorhergehenden Phasen des Entwicklungszyklus (s. Abb. 1) und fehlende Unterstützung von Makro-Strategien. Diese Punkte sind zwar auch bei den anderen Klassen heute noch unbefriedigend, bei diesen kann aber wenigstens mittelfristig auf neue Entwicklungen gehofft werden – bei den allgemeinen Web-Entwicklungswerkzeugen dagegen sind diese Schwächen darin begründet, dass Kursautoren einen wenig relevanten Käuferkreis darstellen.

Versucht man – um ein Beispiel zu nennen – die Entwicklung einfacher „Multiple-Choice"- ITAs mit einem allgemeinen Web-Entwicklungswerkzeug wie FrontPage™, so besteht die angenehme Überraschung darin, dass neuere Versionen hier recht ansprechende Unterstützung bieten. Schlechter sieht es dagegen mit der – für WBT oft gewünschten – automatischen Analyse der Lerner-Eingaben und Erzeugung von Rückmeldungen aus. Hier bleibt entweder der Weg über einen Web-Server – meist nur mit Kommunikation via eMail – oder der Griff zu einer skript- oder programmiersprachlichen Lösung wie Java. Große Unterschiede in dieser Werkzeugklasse betreffen auch die Unterstützung für die konsistente und wartungsfreundliche Handhabung der Menge von Seiten, welche einen Kurs als Ganzes repräsentieren, ggf. in diversen (z.B. Sprach-)Versionen – NetObjects Fusion™ ist hier ein positives Beispiel.

Vorteile allgemeiner Web-Entwicklungswerkzeuge liegen insbesondere in deren Verbreitung. Diese führt zu oftmals moderaten Preisen und insbesondere zum nötigen

Konkurrenzdruck bei gleichzeitigen finanziellen Möglichkeiten für den Hersteller, um mit der rasanten Entwicklung der Web-Technologie Schritt zu halten. Auch die Benutzerfreundlichkeit und gesamte Unterstützung können bei solchen weit verbreiteten Werkzeugen besser sein als bei auf WBT beschränkten. Diese Werkzeugklasse kann daher z.B. in Frage kommen, wenn der Autor a) ohnehin für komplexere Aufgaben den Rückgriff auf eine Programmiersprache vorsieht und b) evtl. Werkzeuge für andere Phasen des Entwicklungszyklus gemäß Abb. 1 verwenden will, welche nicht auf ein bestimmtes Autorenwerkzeug zugeschnitten sind (ein Beispiel hierfür wäre die „Lotus Learning Space“ Plattform).

Spezielle WBT-Autorenwerkzeuge: auf den ersten Blick scheint dies natürlich die Klasse der Wahl zu sein. Diese Klasse ist jedoch die bislang jüngste und umsatzschwächste, worunter Qualität und Leistungsumfang der Werkzeuge oft noch leiden. Der Gegensatz bei allgemeinen Web-Entwicklungswerkzeugen wurde bereits aufgezeigt, die dritte Klasse (CBT-Werkzeuge) kommt diesbezüglich vor allem deshalb besser weg, weil CBT-Werkzeuge häufig bereits viele Jahre länger am Markt sind und der Umsatz am CBT-Markt auch den am WBT-Markt deutlich übersteigt. Zudem können CBT-Werkzeuge, wenn sie WBT als „Vertriebskanal“ unterstützen, die konsistente und wirtschaftliche Entwicklung für verschiedene „Kanäle“ (CD-Rom u.v.a.) unterstützen.

Mangelnde „Reife“ und begrenzter Markt von WBT-Autorenwerkzeugen lassen befürchten, dass diese noch über Jahre hinaus dem HTML-Standard verhaftet bleiben werden, während dieser Artikel davon ausgeht, dass etliche wünschenswerte Funktionen nur auf der Basis der aufkommenden Web-Technologie auf XML-Basis vernünftig realisierbar sind. Zwar stellt die Einbeziehung von Programmiersprachen wie Java auch für diese Werkzeugklasse einen universellen Ausweg dar, damit begibt sich der Autor dann aber in eine geschlossene Welt, wo die pädagogisch-didaktische Unterstützung wieder ebenso fehlt wie die Wiederverwendbarkeit und Austauschbarkeit von Kursmodulen durch Standardisierung und Autorenwerkzeug-seitige Modularisierung.

Dreamweaver Attain™ gehört zur hier besprochenen Klasse. Fortgeschrittene pädagogisch-didaktische Interaktions-ITAs (knowledge objects genannt) kennzeichnen es ebenso als WBT-spezifisch wie die Integration des „Lerner-Entwicklungszyklus“ in sog. knowledge tracks. Die Attain™-Plattform beinhaltet auch eine Reihe von Werkzeugen für den Kurs-Entwicklungszyklus nach Abb. 1. Wie alle in diesem Kapitel besprochenen Werkzeuge ist es allerdings HTML-basiert mit allen Nachteilen, die diesbezüglich im nächsten Kapitel diskutiert werden.

Web-fähige CBT-Autorenwerkzeuge: in dieser Klasse finden sich die meisten der „berühmten“ Werkzeuge wie Authorware™, Toolbook™, Quest™, oder IconAuthor™ – was deutlich macht, dass die entsprechenden Hersteller dem „Vertriebskanal“ Web große Bedeutung beimessen. Unter der Vielzahl von Vertretern dieser Klasse finden sich praktisch alle eingangs erwähnten Metaphern; die oft jahrelange Entwicklungsgeschichte sorgt dafür, dass umfangreiche Funktionalität bereitgestellt wird (z.B. mit einer Vielfalt von ITA-Typen). Zunehmend wird die Integration mit weiteren Werkzeugen des Lebenszyklus vorangetrieben, besonders zeichnet sich die Klasse aber durch gute Einbindung von Servern aus, sowohl für den Online-Zugriff der Lernenden als auch hinsichtlich Feedback und Evaluation. Medienkompression mit Streaming- und Caching-Unterstützung sind vereinzelt sehr weit fortgeschritten.

Das Vorgenannte macht deutlich, dass das Web hier in der Tat hauptsächlich als neuer „Vertriebskanal“ verstanden wird, aber kaum als wirklich neuartiges Medium mit pädagogisch-didaktisch gesehen neuen Möglichkeiten und Randbedingungen. Entsprechend gering ist der „Durchdringungsgrad“ der Werkzeuge mit Web-Technologie. Zwei wesentliche Alternativen können unterschieden werden: bei etlichen Werkzeugen wird HTML quasi als „Rahmen“ verwendet, in das auf dem Umweg über firmenspezifische *Plug-Ins* für Web-Browser (wie Netscape™ oder Internet Explorer™) proprietäre und in sich geschlossene Lösungen eingebettet werden. Bei anderen wird „natives“ HTML als Zielsprache angeboten, aufgrund der Mängel dieser Technologie muss dann aber die Funktionalität gegenüber anderen Vertriebskanälen eingeschränkt werden (was das ursprüngliche Ziel „eine Quelle, mehrere Ziel-Kanäle“ konterkariert) – als Ausweg bleibt teilweise noch Java, mit den bereits diskutierten Nachteilen.

Toolbook™ beispielsweise bietet beide genannten Alternativen an: ein Kurs kann entweder für das sog. Neuron™-Plug-In zugeschnitten werden oder muss für eine kombinierte HTML/Java-Lösung vorbereitet werden, allerdings (zum Zeitpunkt einer durchgeführten Evaluation) mit deutlichen Einschränkungen in der Funktionalität.

Trotz der genannten Nachteile könnte das bisher diskutierte den Anschein erwecken, die „WBT-Welt“ sie durchaus in Ordnung, weil eine große Zahl von leistungsfähigen Werkzeugen bereitsteht und je nach Problem zwischen Alternativen abgewägt werden kann. Die vorliegende Veröffentlichung soll jedoch helfen darzustellen, dass alle genannten Werkzeuge nur (i.w.) eine einzige Klasse möglicher Lernsysteme unterstützt, während für weitere Klassen ein großer Bedarf vorhanden wäre.

Perspektiven Web-Basierter Lernsysteme

Wie erwähnt wird nachfolgend versucht, heutiges WBT kritisch zu hinterfragen und Perspektiven aufzuzeigen. Dabei ist vor allem die Erkenntnis wichtig, dass heutiges WBT nicht mit Computer- und Internet-basiertem Lernen allgemein gleichzusetzen ist. Bezeichnet man Systeme für computergestütztes Lernen als „Lernsysteme“ LS, dann sind fast alle bekannten WBT-Systeme nur einer ganz bestimmten Klasse von LS zuzuordnen, den deskriptiven Lernsystemen DLS– und meist genauer den tutoriellen Lernsystemen (zu beachten ist, dass LS, welche speziell auf Web-Technologie aufbauen, als Web-basierte LS bezeichnet werden; dabei wird innerhalb des allgemeinen Begriffes LS die benutzte Technologie präzisiert, nicht aber die Ausprägung – diese Klassifikation wird für die nachfolgende Diskussion daher erst gegen Ende wieder aufgegriffen). Tabelle 2 ordnet WBT in ein Schema der wesentlichen bekannten LS-Klassen ein. Die getroffene Einteilung wird in der Literatur bisweilen modifiziert verwendet und ist teilweise vereinfachend, für die Zwecke dieses Artikels aber nach Ansicht des Autors gut geeignet. Dabei werden drei Klassen und etliche Unterklassen unterschieden wie nachfolgend sehr kurz (und stark vereinfachend) dargestellt.

Deskriptive Lernsysteme DLS konzentrieren sich auf die (häufig multimediale) Präsentation von Inhalten, „unterbrochen“ durch Interaktionen mit zugehöriger Lernerevaluation und nachfolgender Entscheidung über die weitere Abfolge von Präsentationen. Lernstoff wird also im Wechsel angeboten und geprüft. Das Gros an WBT-Material gehört zur tutoriellen DLS-Klasse; bei „Drill-and-Practice“-DLS stehen Übungsaufgaben stärker im Vordergrund, das Grundprinzip ist jedoch ähnlich.

Modellbasierte Lernsysteme MLS stellen in der Tradition der LS-Forschung eine sehr umfangreiche Klasse dar, in der Literatur oft in grundlegend verschiedene Klassen unterteilt. Für die nachfolgende Diskussion ist es aber hilfreich, hier eine einzige Oberklasse zu verwenden. „Modell“ bezieht sich dabei im wesentlichen auf den Lernenden und/oder den „Gegenstand“ des LS (in erster Näherung die behandelte Wissensdomäne). Bei intelligenten tutoriellen Systemen ITS werden oft anspruchsvolle KI-basierte Modelle (z.B. Expertensysteme) verwendet; historisch gesehen hat sich der Anspruch als überzogen und gefährlich herausgestellt, Modelle der Wissenskonstruktion im kognitiven System (‚Gehirn’) der Lerner zu erstellen, die alle möglichen Irrwege (‚misconceptions’) einbeziehen.

In den vergangenen Jahren wurden daher stärkere Hoffnungen in Simulationen gesetzt; diese MLS-Unterklasse reicht von den ITS-verwandten aber ‚berechenbareren' Mikrowelten (welche sehr gut mathematisch oder formal vollständig fassbar sein müssen wie z.B. eine geschlossene Gruppe physikalischer Gesetze) über Verhaltens- und Anwendungssimulationen (die vor allem „Trockenübungen" für kritische Verhaltensmuster oder ‚teure' Anwendungen bieten), Rollenspiele (welche Rollen aus der realen Welt modellieren wie die des Börsenspekulanten) bis zu echten Spielen (die das vor allem aus Gründen der Motivation eingeführte Spielgeschehen modellieren und die Lernziele hinter den Spielzielen „verstecken").

Bei der dritten MLS-Unterklasse stehen im weitesten Sinne „Programme" sowie zugeordnete „Pläne" im Vordergrund; das LS versucht den Plan zu modellieren, den der Lernende als Programm festlegen will. Programmierumgebungen (als LS-Klasse) stellen meist eine einfache Sprache zur Verfügung (z.B. LOGO) und unterstützen den Lerner ‚nur' beratend bei der Erarbeitung eines Planes sowie dessen Umsetzung, die Experimentierfreiheit für den Lerner ist dabei das gewollte große Plus; Problemlösesysteme beschränken die ‚Programmiersprache' auf vergleichsweise wenige Blöcke oder Bausteine und können daher weitergehende Hilfestellung geben, schränken aber die Wahlfreiheiten für Lerner stark ein. Bei Hilfesystemen liegt fertige Software (z.B. Textverarbeitungssystem) zugrunde, an die Stelle des Programmierens oder des Zusammenstellens von Problemlöse-Bausteinen tritt hier das ‚sinnvolle' Anwenden des Programmes – das ebenfalls als Plan (bzw. Menge möglicher Pläne) modelliert werden kann. Das Hilfesystem hat hierbei meist sehr viel weitergehende Plan-Modelle und bietet Hilfestellung durch Schlüsse von Benutzeraktionen auf vermutete Plan-Varianten.

Werkzeugbasierte Lernsysteme WLS zielen auf sehr stark selbstorganisierte Lernformen und stehen damit im krassen Gegensatz zu DLS und MLS, die als konservative (DLS) bzw. anspruchsvolle (MLS) Systeme verstanden werden können, bei denen das LS die ‚Initiative' besitzt. Auch der Gruppenaspekt (kooperatives Lernen) wird bei WLS stärker betont. Sollen die Lernenden eine Wissensdomäne selbst systematisch aufarbeiten (strukturieren, evaluieren), so eignen sich kognitive Werkzeuge. Sollen sie vor allem kooperieren, dann bietet sich der Einsatz von Groupware an. Falls auch die Organisation des Lernprozesses selbst von den Lernenden (mit)bestimmt werden soll, können Werkzeuge zur Arbeitsorganisation eingesetzt werden. Der hohe Freiheits- und Selbstbestimmungsgrad für die Lernenden ermöglicht bei all diesen Werkzeugen einerseits in höchstem Maße aktives Lernen, andererseits sind systemseitige Hilfestellungen z.B. bei unbefriedigendem Verlauf (gar auf Basis von Lernermodellen) sehr schwierig.

Tabelle: Klassen von Lernsystemen LS

LS Klasse		Unterklasse	Stärken	Schwächen
deskriptives Lernsystem DLS		tutorielles LS	kommerzieller Erfolg (WBT)	fehlt: Modelle, Kooperation, Exploration
		drill&pract.	Übung	
Modellbasiertes Lernsystem MLS	ITS		Lernermodell Domänenmod.	Aufwand hoch, kaum Wiederverw.
	Simulation	Rollenspiel	Modelle, Interaktion	Web-Technologie nicht genutzt, keine Standards, kaum Wiederverwendung
		Spiel	motivational	
		V/A simulat.	Trockenversuch	
		Mikrowelt	„exakt“, KI	
	Programmierung	Progr.-Umgebung	Kreativität + Freiheiten	wie oben, plus: Kooperation & Offenheit schwierig
		Problemlösesystem	Kreativität + Modelle	
		Hilfesystem	Just-in-time, (+Learnermod.)	
Werkzeugbasiertes Lerns. WLS		kognitives Werkzeug	konzeptuelles Verständnis	Einbindung von Lernermodell & ‘Hilfen’ schwierig
		Groupware	kooperativ	
		Arbeitsorg.-Werkzeug	selbstorg. Lernen	

Pädagogisch-didaktische Bewertung: ohne im gegebenen Rahmen die notwendigen Differenzierungen vornehmen zu können, sei kurz auf die Eignung der genannten LS-Klassen eingegangen. Die seit Jahren diskutierte Verlagerung von behavioristischen zu kognitivistischen Lerntheorien und von instruktivistischen zu konstruktivistischen pädagogischen Philosophien [4, 5, 8, 11] schlägt sich nicht zuletzt darin nieder, dass im akademischen Umfeld MLS im Zentrum des Interesses stehen, während DLS eher als Relikt der späten 60er Jahre betrachtet werden; vor allem der konstruktivistische Ansatz wird durch WLS sogar noch stärker reflektiert als durch MLS.

Darüber hinaus hat aber gerade die rasante technologische Entwicklung (nicht zuletzt von Web und Internet) die Forderung verstärkt, Lehrende sollten sich vom (heute oft unmöglichen) alles wissenden und bestimmenden „sage on the stage“ zum bescheideneren aber wirkungsvolleren „guide on the side“ entwickeln, die den Lernvorgang als Coach begleiten statt bestimmen. Das entspricht auch den Anforderungen des Berufslebens, wo vermehrt sogenannte Softskills verlangt werden wie die Fähigkeit zu selbst-

organisierter Teamarbeit und zum „Knowledge-Working“ als mühsamer Informationsbeschaffung und –aufbereitung in den Flutwellen des Internet (Stichwort „Information Overload“); auch hier sind vor allem WLS, teilweise noch MLS, aber keineswegs die „Information optimal vorverdauenden“ multimedialen DLS angebracht.

Allein durch „schöne Multimediapräsentationen“ werden DLS auch dem Ruf nach Motivation wesentlich schlechter gerecht als z.B. spannende Spiele oder andere stark interaktive MLS [9]. Auch bezüglich Motivation kann das Web im Kontext von WLS zunehmend eine wichtige Rolle spielen, da immer mehr „reale“ Vorgänge (z.B. des Wirtschaftslebens) im Web stattfinden und so authentische statt künstliche Übungsfelder im Web (über WLS) zugänglich gemacht werden können.

Ob man also die pädagogisch-didaktischen Theorien betrachtet, die veränderten Anforderungen aus der beruflichen Realität oder den Aspekt Motivation, fast immer erweisen sich MLS als den DLS vorzuziehen und WLS, vor allem Web-basierte WLS, als noch vorteilhafter. DLS bleiben natürlich für klar begrenzte Teilaufgaben sinnvoll, vor allem dort, wo kompakt elementare Zusammenhänge im klassischen Sinn „vermittelt“ werden müssen. Allerdings bleibt bei MLS das Problem des erheblichen Entwicklungsaufwandes (teilweise auch der überzogenen Ansprüche) und bei WLS das Problem der schwierigen Hilfestellungen.

Schlussfolgerungen zur Web-Technologie

Die oben diskutierten Vorzüge von WLS waren großteils an den Bezug zum Web gebunden, Web-basierte WLS stellen also eine der wichtigsten wünschenswerten LS-Klassen dar. MLS wurden bislang noch kaum für das Web entwickelt, noch weniger auf der Basis „nativer“ Web-Technologie. Das verwundert auf den ersten Blick, weil viele erfolgreiche MLS-Entwicklungen auf der Basis von Hypertext- bzw. Hypermedia-Systemen durchgeführt wurden und weil das Web ja ein solches System darstellt (vgl. http als ‚hypertext transfer protocol’). Bei näherer Analyse zeigt sich, dass die heutige, HTML-basierte Web-Technologie viele Fähigkeiten anderer Hypermedia-Systeme vermissen lässt. Hoffnung bietet die aufkommende, auf XML und verwandten Standards basierte Generation, wie an einigen Beispielen aufgezeigt werden soll:

- Wie das „T“ in HTML andeutet, ist das heutige Web trotz eingebundener Medien Text-zentriert, außerdem stellen HTML-Seiten eigenständige Dokumente dar, während die Kopplung zwischen solchen Seiten (über URLs) relativ lose ist. Auch Werkzeuge sind auf das Konzept der HTML-Seite zentriert. Klassischerweise wird aber Hypermedia als Ansammlung atomarer (medialer) Knoten und Links verstanden. Ein solches Modell lässt sich (wenn auch nicht gerade trivial) in XML realisieren.
- Mittels XML, Xlink/Xpointer und RDF [1, 2, 7] lassen sich anwendungsspezifische Knoten- und Link-Typen realisieren, eine der Grundvoraussetzungen für fortgeschrittene Funktionalität und Wiederverwendbarkeit – HTML bietet hier so gut wie keine Unterstützung.
- Weitere Beispiele für Defizite der heutigen und Möglichkeiten der künftigen Generation von Web-Technologie sind: die Trennung von (ggf. mehreren Ebenen von) Knoten-/Link-Strukturen und den eigentlichen Inhalten; die konsistente Verwaltung dieser Strukturen (z.B. ohne ‚Error-404’-Meldungen für nicht mehr existierende Links) sowie (z.B. Benutzermodell-basierte) Lernsystem-gesteuerte Benutzerführung durch die Links; die echte Einbeziehung von Multimedia-Präsentationen und –Synchronisationen, wobei deren Bestandteile Knoten der „nativen“ Web-Technologie sind (hier bietet das XML-basierte SMIL [10] eine Abkehr von prorpietären, abgekapselten Lösungen wie Shockwave™).

Neben dieser Liste, die noch lange fortgesetzt werden könnte, zeigen auch Arbeiten zu Ontologien [6] und Standards wie das XML-basierte *Learning Objects Model (LOM)* [3], dass von der kommenden Generation von Web-Technologien wichtige Beiträge zu den wesentlichen in diesem Beitrag geforderten Entwicklungen erhofft werden können: dem Wandel von fast ausschließlich DLS zu mehr MLS und vor allem WLS im Web sowie die Bereitstellung von Werkzeugen, damit diese mit vertretbarem Aufwand und hohem Wiederverwendbarkeitsgrad entwickelt werden können. XML und verwandte Standards ermöglichen zwar diese Entwicklungen, garantieren sie aber keineswegs automatisch. Vielleicht kann die vorliegende Veröffentlichung diesbezüglich einen kleinen Beitrag leisten, um das Bewusstsein der Entwickler zu stärken und den Druck der LS-Entwickler auf die Werkzeughersteller zu erhöhen.

Literatur

1. R. Daniel, S. DeRose, E. Maler, XML Pointer Language (XPointer) V1.0. W3C Recommendation, June 2000. (See http://www.w3.org/TR/xptr.)

2. S. DeRose et al., XML Linking Language (XLink) 1.0, W3C Recommendation July 2000, (See www.w3.org/TR/xlink/)

3. Draft Standard for Learning Object Metadata, IEEE Learning Technology Standardization Committee, 3/2000 (See http://ltsc.ieee.org/doc/wg12/LOMv4.1.pdf)

4. I. Harel, S. Papert (Eds.). Constructionism, Norwood, NJ: Ablex, 1991.

5. D. Jonassen (Ed.), Handbook of Research for Educational Communications and Technology. Macmillan, New York, 1996:

6. M. Klein et al. "The Relation between Ontologies and Schema-Languages: Translating OIL-Specifications to XML-Schema" In: Proc. Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI-00, Berlin, Germany Aug. 20-25, 2000.

7. O. Lassila, "RDF Web Metadata: A Matter of Semantics", IEEE Internet Computing, Vol. 2, No. 4, July/Aug. 1998

8. T. Reeves, "A Research Agenda for Interactive Learning in the New Millennium", Proc. Ed-Media '99, Seattle, WA, USA; AACE, Charlotteville, VA, 1999.

9. L. Rieber, "Seriously considering play: Designing interactive learning environments on the blending of microworlds, simulations, and games", Educational Technology Research & Development, Vol. 44, No. 2, 1996, pp. 43-58.

10. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C Recommendation June 1998 (See http://www.w3.org/TR/REC-smil/)

11. B. Wilson, M. Lowry, "Constructivist Learning on the Web", in Liz Burge (Ed.), Learning Technologies: Reflective and Strategic Thinking. Jossey-Bass, San Francisco, 2001. http://ceo.cudenver.edu/~brent_wilson/WebLearning.html