Knowledge Management Tools for Instructional Design

☐ J. Michael Spector

Advances in computer technology typically find their way into education after a short generation of success in other settings. This is an elaboration of one such technology—knowledge management systems (KMS)—and its application to instructional design. An examination of the development of KMS from information systems, computer-supported collaborative work environments and object-oriented systems, leads to a discussion of reusability. The focus is on the use of KMS by instructional designers. A conceptual framework for distributed instructional design is provided along with examples of support tools. These tools and the associated design framework are in use, and anecdotal evidence of effects and impact is provided. As such tools become more widely used to support the planning, implementation and management of instructional systems and learning environments, it is reasonable to expect the nature of instructional design practice to change.

☐ There are many educational research and technology projects reporting a variety of outcomes and lessons learned with regard to effective integration of technology into learning and instruction (Dijkstra, Seel, Schott, & Tennyson, 1997; Spector, 1994, 1995; Spector & Anderson, 2000). What can we learn from these projects and experiences? Is there a clear and coherent instructional design framework for technology enhanced learning environments? What are the most promising approaches to instructional design? Are there particular tools that can assist? What kinds of evaluations will insure that the process of designing such environments will become progressively more effective?

The purpose of this paper is to suggest answers to these questions. Technology integration in education is not a new concern. Moreover, there are a number of dimensions to technology integration. Its most obvious dimension in education is arguably that of instructional delivery in the form of advanced learning environments (e.g., simulators, virtual worlds, etc.). The dimension of concern in this paper, however, is that of instructional design. Instructional design itself is a large area ranging from the assessment of needs and analysis of requirements through the planning and elaboration of instructional solutions. The focus here is on the impact of a new technology—knowledge management systems (KMS)—on instructional planning.

A KMS can be described as an integrated collection of tools. Indeed, this is the typical depiction of KMS in the computing literature. The next section will provide a historical perspective with regard to the development of key knowledge management tools. These tools provide support for collaborative work, object orientation, and reusability, each of which is discussed briefly. The subsequent section provides a description of KMS that goes beyond a narrow tool perspective and includes the user as an essential part of the system. This is followed by an example of the use of such a system to support instructional design. The conclusions drawn from this discussion are not totally encouraging, however. While there is strong potential to use knowledge management technology to improve instructional design, the reality of instructional design practice suggests that this potential may not be fully realized for these reasons: (a) competing instructional design firms are not likely to openly share learning objects and corporate knowledge; (b) instructional designers tend to believe that instructional decision making is best left to human experts; and, (c) some educational theorists who advocate completely open-ended learning and discovery environments believe that instructional design has no place in education (for an elaboration, see Spector, 1995).

A BRIEF HISTORY OF COMPUTING AND INSTRUCTIONAL SYSTEMS

One way to characterize the development of computer systems is in terms of generations of languages and systems. Computer languages are often represented as having progressed from early machine-oriented code (bit-level or hexadecimal representations of specific machine instructions) to higher level and more abstract representations.

Computer languages have clearly become more abstract, more distant from specific machine-level concerns and more oriented at problem solving from the user's perspective. The conclusion that is enticing is that modern computer languages make it possible for subject experts without special training in software engineering to create effective computer programs to solve relevant classes of problems. This has not happened to the extent predicted. As computer languages have grown further from the machine perspective and closer to the human problem-solver's perspective, there has not been such a dramatic increase in the accessibility of computer programming to larger and larger groups of nontechnically trained problem solvers. In short, the reusability potential of object oriented programming remains limited to experts (Hurwitz, 1997).

However, the ability of nonspecialists to make effective use of computers has grown in other ways. As experts at research and development laboratories were working on new programming developments to solve large-scale problems, they found the need to develop associated tools to facilitate their work. These associated tools provided support for: (a) the ability to pass notes efficiently back and forth (computer-facilitated communication); (b) the ability to schedule meetings and circulate notices and agendas quickly and efficiently (computer-facilitated coordination); (c) the ability to share and exchange working documents and artifacts (computer-facilitated collaboration), and (d) the ability to automatically track and audit multiple versions of various artifacts (computer-facilitated control). In short, the foundation was laid for what evolved into integrated tool sets to support collaborative work and eventually to support enterprise-level activities.

Computer-Supported Collaborative Work

Software engineers developed a range of computer-supported systems to facilitate software development processes. Such tools were created to promote effective teamwork on complex and large-scale efforts and have found use in other settings where they are commonly referred to as computer-supported collaborative work (CSCW; Wilson, 1991). These new technologies allow for creating environments and processes that support instructional design activities in a distributed setting as an essential part of refining the routine.

The term *computer-supported collaborative work* can be traced to 1984 (Grief, 1988). Computer scientists used CSCW for an invited workshop focused on the development of computer systems to support people in various work-related activities (Bannon, 1991; Bannon & Schmidt, 1991). CSCW systems are generally created and customized to support multiple people working at the same location or at different locations connected by a network. The orientation in a CSCW system is not on a particular task but rather on the need for different persons, typically at different locations and possibly at different times, to work together in creating various artifacts aimed at solving some kind of problem (Ganesan, Edmonds, & Spector, 2001; Koschmann, 1996).

In short, the typical purpose of a CSCW system is to provide an environment that supports workplace collaboration and instantiates support for distributed cognition (Salomon, 1988, 1992, 1993). Early tools that evolved into key elements of a CSCW system include e-mail, computer-based calendars, and electronic bulletin boards (Wooley, 1994). These tools led to the development of integrated platforms for CSCW applications (e.g., Lotus Notes, Xerox DocuShare, Seven Mountains Integrate / Aspire, etc.) that include more elaborate forms of support, such as:

- Dynamic support for groups and subgroups;
- Variable and adaptable interfaces;
- Synchronized control of artifacts and communications;
- Communication and coordination among groups and subgroups;
- Shared and shareable information spaces, and
- Support for increasingly diverse types of artifacts.

As CSCW systems found additional uses in a wider variety of settings, some researchers (notably those arguing for a more constructivist perspective) began to explore the efficacy of collaboration in various problem-solving and learning settings (Jonassen, Hernandez-Serrano, & Choi, 2000; Scott, Cole, & Engel, 1992). One complex task in the domain of systems dynamics is model building. An early example of online group support for complex tasks is group model building (Vennix, 1996). In systems dynamics, then, tools and techniques were developed to foster the process of group-model building (Morecroft & Sterman, 1994; Richardson & Andersen, 1995; Vennix, 1996). Model building tools and techniques support group decision making, small-group communication, and project management. The group model-building process and techniques may also inform instructional planning and analysis, another complex task or collection of tasks, and activities (van Merriënboer, 1997). An added advantage of collaborative group work is that participation in group planning processes is easily extended to support user-centered and participatory design.

In summary, CSCW systems are not typically aimed at support for a particular task or activity. Rather, they are developed to support group work on complex activities (Dillenbourg, Baker, Blaye, & O'Malley, 1996; Koschmann, 1996). The central purpose is to facilitate group work. Such systems are clearly appropriate for instructional design and development. Indeed, many instructional design projects and efforts now use such tools, including the ADAPT^{IT} Project (de Croock, Paas, Schlanbusch, & van Merriënboer, this issue; Spector, Eseryel, & Schuver-van Blanken, 2001). Such systems do not directly address object orientation and reusability, two of the key notions in the evolution of programming languages, which are briefly reviewed prior to the discussion of KMS.

Object Orientation

Object orientation is derived from the evolution of programming languages. Basically, the notion is that rather than think in terms of data structures and machine operations, programmers should be encouraged to think in terms of things that have direct and obvious parallels in the real world to be supported by computer applications. Approaching problems this way should in principle promote flexible, generalizable, and reusable solutions to recurring problems. The real world does not come packaged as a collection of mathematical functions and data types. Rather, the real world consists of things—objects—which are acted on by other things and react in various ways. Similar things tend to behave in similar ways (a hint at reusability). A designer can create new things by indicating the kind of thing to be created, which brings together a family of features or collection of predefined characteristics (including behaviors or methods). These inherited characteristics can of course be modified.

An example of a reusable object from computing is a checkbox. Software engineers often have a requirement for an interface to display a choice, record the user's decision, and then report that decision to the control system so that appropriate action can occur. This is a recurring requirement and the code is highly reusable. Rather than program a checkbox each time the requirement occurs, the software engineer can obtain a precoded object from a library of objects and simply indicate the details pertinent to this instance (e.g., which decisions are allowable, where to branch for each decision, etc.).

Similarly, instructional designers encounter recurring requirements that can be served by the use of resusable learning objects (Wiley, 2001). An example of a reusable learning object is an object that "teaches" a simple concept at an introductory level. This is a requirement that instructional designers frequently encounter and there is a well established pedagogy to support such teaching: present a definition, an example, a nonexample, and an opportunity to practice, and test the application of the concept with novel examples and nonexamples (Merrill, 1993).

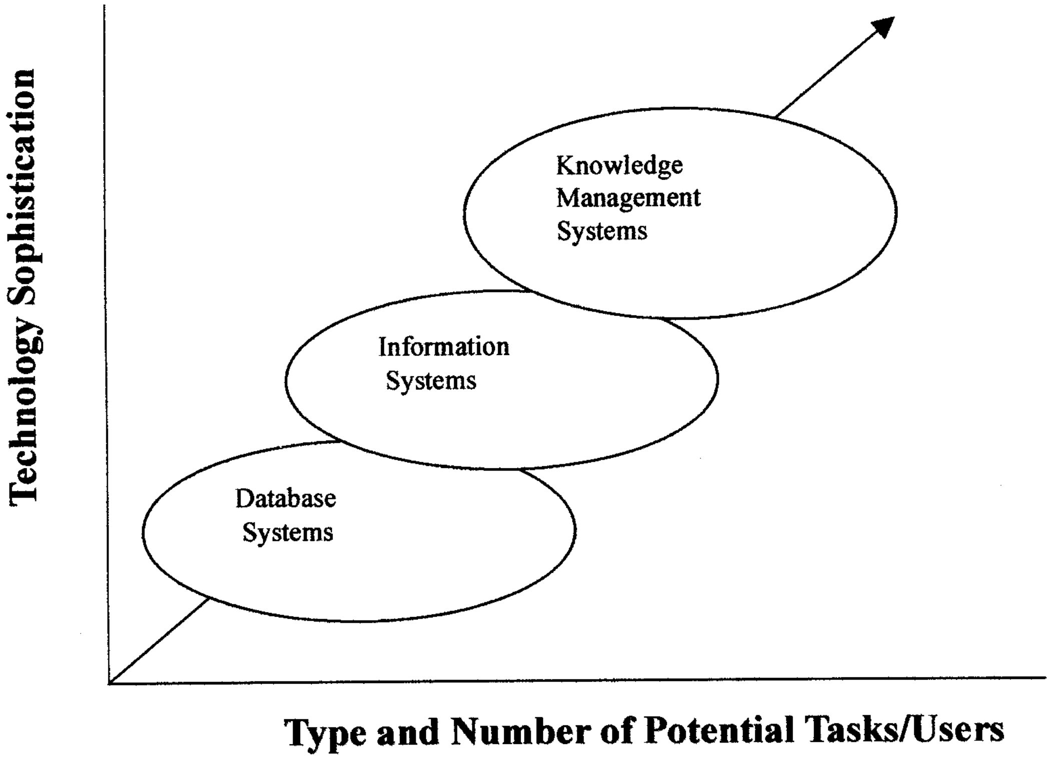
Object orientation puts ontology first and fits extremely well with one well-established instructional design ontology, namely Merrill's (1993) Second Generation Instructional Design—ID^2. Merrill's world consists of objects (abstract and concrete entities), activities (things people do), and processes (things that occur in various objects and situations apart from human activities). Instructional development systems built with an explicit ontology such as Merrill's ID^2 represent an application of object orientation in the domain of instructional design (Spector, 1999). That such an approach can enhance productivity and instructional quality within a development team has been demonstrated and argued strongly by Merrill (1993). Whether such object orientation results in reusability outside the development team or enterprise is a different matter because such reuse would require knowing details about such knowledge objects and having ready and flexible access to them.

A second kind of object orientation is becoming prominent, namely Web-based knowledge objects for instructional purposes (Merrill, 1998; Wiley, 2001). Such use is primarily based on a derivative of standard generalized markup language (SGML)—the predecessor to hyptertext markup language (HTML). This relatively recent effort is called the extensible markup language (XML; see http://www.w3.org/TR/-xhtml1/ for additional details). Basically, XML regains much of the flexibility and power of SGML while maintaining the familiarity of HTML for users. In XML, it is relatively easy to introduce new elements or additional element attributes making the language extensible. When XML is combined with the notion of metadata-defined learning objects, the potential for distributed reuse of knowledge objects for instructional uses becomes real. SCORM (shareable content object reference module) represents just that reusability technology for distributed learning environments (see http://www.adlnet.org/Scorm/scorm_index.cfm for additional information).

Reusability

In order to realize the potential of reusability, there are two essential aspects: (a) Technologies such as XML and SCORM represent only one of these essential aspects—the enabling underlying technology aspect; and (b) the human use of such enabling technologies is the second aspect. An organization can conceivably devote considerable resources to developing a repository of objects with appropriate metadata tags to indicate the type of learning object involved. However, such resources will go largely unused unless instructional designers and developers are properly trained. Moreover, unless and until such resources are shared across enterprises and among institutions, the potential of reusable learning objects and instructional metatagging will not be realized. As suggested earlier, failure to follow through on the human use side of such technologies can result in suboptimal outcomes (Brooks, 1995). In an important sense, reusability is not fundamentally about metadata or object orientation. Rather, it is a human use issue that

Figure 1 ☐ Evolution of knowledge management systems (KMS).


Technology Sophistication
Knowledge Management Systems
Information Systems
Database Systems
Type and Number of Potential Tasks/Users

sinks or floats on perceptions, proper pretraining, ongoing support, incentives for collaboration and sharing, and so on. The institutional support climate for collaboration and personalities are crucial for systematic success with regard to reuse.

KNOWLEDGE MANAGEMENT SYSTEMS

KMSs have evolved from earlier database and information management systems (see Figure 1). Modern problem- and object-oriented programming languages evolved from earlier machine-oriented languages. Modern KMSs support multiple users performing a variety of tasks in a flexible and dynamic manner, and have evolved from earlier user- or task-oriented systems. Databases were a relatively early development in the enterprise use of computers. Early databases were simple collections of records composed of specific fields that could support specific users performing specific tasks (e.g., a person searching a database to see how many of a certain item are available). Databases became widely used to support a variety of users with differing requirements. Subsets and supersets of existing databases were created as new users and uses were found. Soon databases were being created that contained redundant data. In some cases only one set of data was maintained.

This led to advances in database technology that included both relational and object-oriented databases aimed at promoting reuse of information and minimizing redundant and unreliable data in various information repositories. Other features were then integrated into databases (e.g., exporting records to a word processor, creating differing access privileges for different users, linking databases to spreadsheets and other enterprise documents, managing projects using information from databases, etc.). These systems with their added functionality beyond simple database management (e.g., add, modify, delete, search, browse) became known as information management systems.

Information management systems, then, have evolved into KMSs as still more features have been added and integrated. Critical features of a KMS include explicit support for:

1. Communication (e.g., e-mail, bulletin boards, group messaging);
2. Coordination (e.g., shareable calendars, groups tasking, etc.);
3. Collaboration (e.g., shareable artifacts, shared work spaces, etc.); and
4. Control (e.g., version and configuration control, audit trails, document locking, etc.).

All of these capabilities support groups working on complex tasks (Kling, 1991; Malone & Crowston, 1993). As already argued, instructional design represents a collection of complex tasks and activities typically accomplished by multiple individuals working on different aspects at different times and perhaps in different locations. In short, the potential for KMSs to impact the work of instructional design groups is quite significant. That KMSs are already being used and have an effect on instructional development groups is a reality (Ganesan et al., 2001). The next section illustrates one such case.

AN EXAMPLE OF KNOWLEDGE MANAGEMENT IN INSTRUCTIONAL DESIGN

Managing large volumes of information and knowledge assets is central to large instructional design or development efforts and is generally well supported by a KMS. Several KMSs are being used to support the collaborative design of instruction, including Lotus Notes (an outgrowth of the early PLATO system), Xerox Corporation's DocuShare & Flowport and SevenMountains's 7M Enterprise (Integrate & Aspire) (Ganesan et al., 2001). The example elaborated here involves DocuShare–Flowport.

DocuShare is a Web-based document management system that is well suited to collaborative group work. DocuShare requires users to have access to and basic familiarity with the Internet. It is a powerful but simple tool that allows the sharing of documents via the Internet. A user can add, post, change, search for, and retrieve information in a secure, controlled environment. Users can exchange documents and multimedia files—any format distributable and accessible via the Web—without requiring users to have any special knowledge or expertise with regard to HTML and other underlying technologies. There are four ways to put documents into DocuShare collections: (a) through a Web browser by clicking ADD FILE; (b) by dragging and dropping files to a networked folder on the user's desktop; (c) from within a word processor or other software integrated into DocuShare by choosing SAVE and indicating a DocuShare collection; and, (d) by scanning documents directly to DocuShare collections using a networked scanner and additional software called FlowPort.

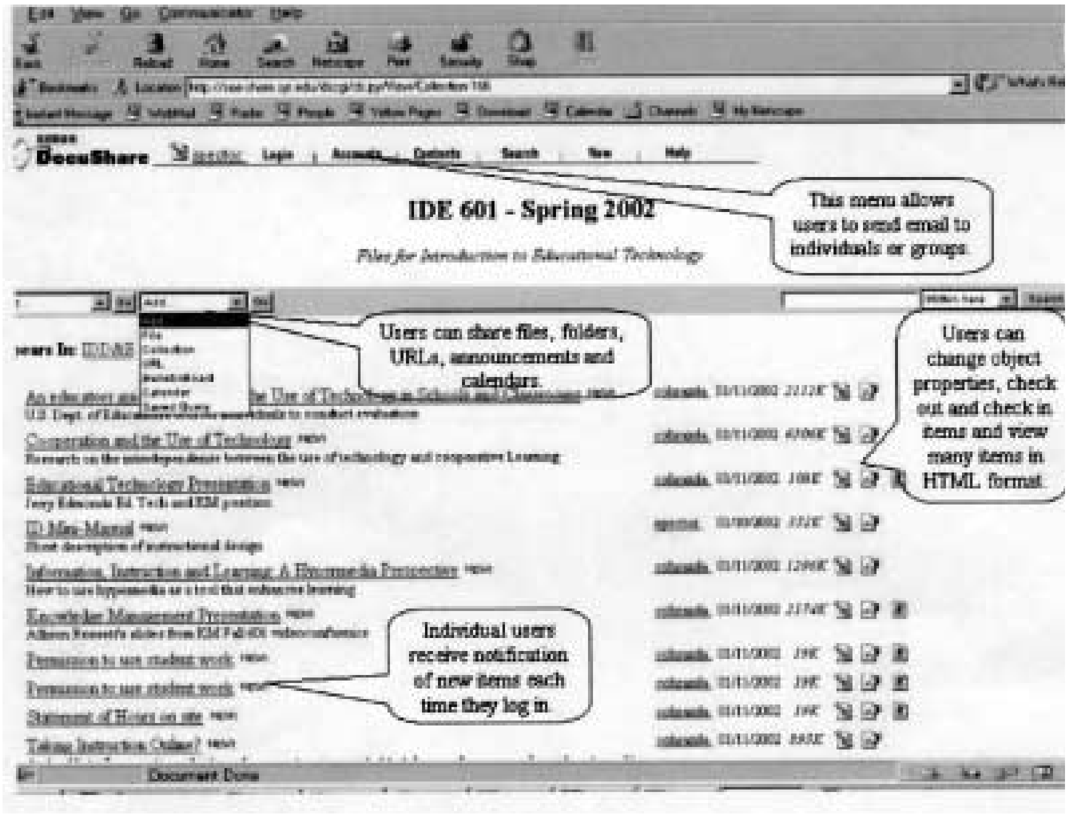
In addition to explicit and strong support for collaborative group work, DocuShare provides support for the other critical aspects of a KMS, as indicated in Table 1.

Table 1 ☐ Docushare support for KMS.

KMS Aspect	*Docushare Support*
Communciation	Bulletin boards for specific collections, e-mail for individuals and groups
Coordination	Calendars for collections with access for both individuals and groups
Collaboration	Shareable documents with locking features to insure document integrity
Control	Automatic versioning of documents, ability to revert to earlier versions, access control by individual, group, document, and collection

Design teams can use DocuShare to manage all of the documents associated with an instructional design or development project. A collection (a directory or folder) is created for the project; users are then given a username and password along with access to appropriate collections. A project can be divided into manageable pieces for various team members with

Figure 2 ☐ DocuShare support for collaborative course collections.


DocuShare
IDE 601 - Spring 2002
Files for Introduction to Educational Technology
This menu allows users to send email to individuals or groups.
Users can share files, folders, URLs, announcements and calendars.
Users can change object properties, check out and check in items and view many items in HTML format.
Individual users receive notification of new items each time they log in.
Document: Done

different roles and access privileges. Team members can create bulletin boards, send messages, and add documents to the various collections. All of the documents and communications are stored on a central server, relieving individual members from worrying about backups and other purely administrative tasks. When a team member checks out a document, DocuShare "locks" the original to prevent parallel changes while another team member is adding or editing material. When a document is checked back into a collection, the lock is removed and other team members can check out and modify the document.

The training required to use DocuShare is minimal—many users are able to make effective use of DocuShare after a simple five-minute starter session or with the assistance of a one-page starter guide. The program does contain a built-in tutorial as well as a variety of job aids, along with short instructions and a longer user's manual. DocuShare has been used to host both online and hybrid courses, and is especially well suited to support front-end planning and document-sharing systems for groups of instructors and designers working together on a variety of courses (Ganesan et al., 2001).

Although not intended to be a Web course management system, DocuShare has been used at Syracuse University to host online courses. There is no inherent support for threaded discussions, but the bulletin feature does support asynchronous discussions. A separate chat program was used to supplement the kinds of interactions commonly supported in Web-based courses.

At present, DocuShare is being used to support the design and development of online courses that are delivered using other Web course management systems, such as BlackBoard and WebCT. Since the courses change each semester and are taught by different instructors, the documents used to support an online course are kept in DocuShare. They can be and have been codeveloped by teams of instructors with DocuShare maintaining version control, allowing the documents to retain meaningful names and allowing designers to revert to earlier versions at any time. Syracuse University uses a variety of Web delivery environments (primarily BlackBoard and WebCT), and some courses have migrated from one environment to the other; using DocuShare as the common document repository has greatly facilitated this process. BlackBoard allows a portion of a course to be made publicly accessible while keeping the remainder open only to registered students. This feature is not currently supported in WebCT but the same effect is easily achieved through the guest account feature of DocuShare.

DocuShare is also being used to support classroom-based courses and as a common repository for course documents regardless of the delivery environment. Overall, the use of DocuShare to support the design, development, and delivery of courses has been positive. Usage has grown somewhat slowly, as would be expected for a new technology. The potential to improve course development and help maintain course consistency and quality is immediately obvious to those who have adopted the system in one form or another. The number of people using DocuShare for educational support in the School of Education has jumped from 2 to almost 20 in less than a year; many more are expected to adopt DocuShare as an instructional development tool as those who have used it spread the word about its ease of use. The real power of DocuShare as a collaborative design and development environment is only now becoming obvious.

CONCLUSIONS

Technology certainly can and does change the way we live and learn. It is also true that many technologies have been oversold in terms of promises of radically improved productivity and fundamental or systemic reform of underlying processes. Both of these general observations are certainly true of educational technology and likely to prove true of the integration of KMSs into the way that instructional design teams work.

Nevertheless, the potential to improve the quality of instructional design or development efforts by having ready and flexible access to existing documents and by involving team members in sustainable collaborative efforts over time and across projects is a reality to be taken seriously about the potential for KMS in instructional design. Unlike the limitations likely to be encountered in making use of metadata (XMS) to make significant improvements in reusability, KMS addresses a wider range of issues and is inherently a human-centered technology. The success of efforts such as the case illustrated here, the ADAPT^{IT} case, and other ongoing efforts suggests that KMS has a bright future in instructional design.

Whether or not a system such as DocuShare is adopted as the underlying engine for collaboration is not the key issue. Lotus Notes is a more widely used KMS with a larger installed base in university settings. The same key features (communication, coordination, collaboration and control) can be found. Lotus Notes, perhaps, has a longer learning curve, but it also has the advantage of having an underlying database engine (Lotus Domino). Seven-Mountains 7M Enterprise is also easy to use and intuitive (much like DocuShare) and has most of the same key features, except for communication, which is not an inherent part of Enterprise but is easily available through e-mail and other commonly installed communications tools. The success of reusability will depend on human use issues, such as the existence of a system that promotes human interaction with shared artifacts without increasing the complexity of the work or adding to the cognitive load placed on individuals. In short, the key to successful reuse is not a particular tagging scheme or a particular technology—the key to successful reuse is in getting people with relevant interests, expertise and motivation to collaborate in ways that obviously extend and enhance what they might accomplish individually.

As instructional designers and developers begin to use these KMSs, it will be worthwhile to study interaction patterns and perceptions about the relative merits of collaborative work. There are many small issues yet to be resolved, such as how to facilitate alternative design or development paths using the same underlying set of documents and artifacts. KMS technology should allow what-if analysis of alternative designs and implementations with reversion to earlier versions or to mixed versions. This kind of hybrid instructional prototyping has never been pursued, although it is a common design methodology in other settings (e.g., airplane and automobile design).

There are also large issues yet to be resolved, such as which designs are likely to achieve desired outcomes efficiently. By supporting the development of multiple versions of a course from a common document repository, researchers will be able to develop greater confidence in which input variables really account for different outcomes. The use of knowledge management tools in instructional design can improve the quality of instruction and add to what we know about the relationship between the design of instruction and learning outcomes. The future of KMS in instructional design appears bright. □

J. Michael Spector [spector@syr.edu], Development Editor-Elect of ETR&D, is Professor and Chair, Instructional Design, Development and Evaluation, Syracuse University, 330 Huntington Hall, Syracuse, NY 13244.

REFERENCES

Bannon, L.J. (1991). From human factors to human actors: The role of psychology and human-computer interaction studies in system design. In J. Greenbaum & M. Kyng (Eds.), *Design at work: Cooperative design of computer systems* (pp. 25–44). Hillsdale, NJ: Lawrence Erlbaum Associates.

Bannon, L., & Schmidt, K. (1991). CSCW: Four characters in search of a context. In J. Bowers & S. Benford (Eds.), *Studies in computer supported cooperative work: Theory, practice and design* (pp. 3–16). Amsterdam, The Netherlands: Elsevier North-Holland.

Brooks, F.P. (1995). *The mythical man-month (20th ed.). New York: Addison Wesley.*

Dijkstra, S., Seel, N., Schott, F., & Tennyson, R.D. (Eds.). (1997). *Instructional design: International perspectives.* Mahwah, NJ: Lawrence Erlbaum Associates.

Dillenbourg, P., Baker, M., Blaye, A., & O'Malley, C. (1996). The evolution of research on collaborative learning. In P. Reimann & H. Spada (Eds.), *Learning in humans and machines: Towards an interdisciplinary learning science* (pp. 189– 211). London: Pergamon Press.

Ganesan, R., Edmonds, G.S., & Spector, J.M. (2001). The changing nature of instructional design for networked learning. In C. Jones & C. Steeples (Eds.), *Networked learning in higher education* (pp. 93–109). Berlin, Germany: Springer-Verlag.

Grief, I. (Ed.). (1988). *Computer-supported cooperative work: A book of readings.* San Mateo, CA: Morgan Kaufmann.

Hurwitz, J. (1997). Component-based development. *Database Management Systems* (June, 1997), 10–12.

Jonassen, D.H., Hernandez-Serrano, J., & Choi, I. (2000). Integrating constructivism and learning technologies. In J.M. Spector & T.M. Anderson (Eds.), *Integrated and holistic perspectives on learning, instruction and technology: Understanding complexity* (pp. 103–128). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Kling, R., (1991). Cooperation, coordination and control in computer-supported work. *Communications of the ACM, 34*(12), 83–88.

Koschmann , T. (1996). Paradigm shifts and instructional technology: An introduction. In T. Koschmann (Ed.), *CSCL: Theory and practice of an emerging paradigm* (pp. 1–23). Mahwah, NJ: Lawrence Erlbaum Associates.

Malone, T., & Crowston, K. (1993). The interdisciplinary study of coordination. *Computing Surveys, 26*(1), 87–119.

Merrill, M.D. (1993). An integrated model for automating instructional design and delivery. In J.M. Spector, M.C. Polson, & D.J. Muraida (Eds.), *Automating instructional design: Concepts and issues* (pp. 157–190). Englewood Cliffs, NJ: Educational Technology Publications.

Merrill, M.D. (1998). Knowledge objects. *CBT Solutions* (March / April), 1–11.

Morecroft, D.W., & Sterman, J.D. (Eds.). (1994). *Modeling for learning organizations.* Portland, OR: Productivity Press.

Richardson, G.P., & Andersen, D.F. (1995). Teamwork in group model-building. *System Dynamics Review, 11*(2), 113–137.

Salomon, G. (1988). *AI in reverse: Computer tools that become cognitive.* Invited address at the American Educational Research Association (AERA). March, 1988. New Orleans, LA.

Salomon, G. (1992). What does the design of effective CSCL require and how do we study its effects? *SIGCUE Outlook—Special Issue on CSCL, 21*(3), 62–68.

Salomon, G. (1993) (Ed.). *Distributed cognitions: Psychological and educational considerations.* New York, NY: Cambridge University Press.

Scott, T., Cole, M., & Engel, M. (1992). Computers and education: A cultural constructivist perspective.

Review of Research in Education, 18, 191–251.

Spector, J.M. (1994). Integrating instructional science, learning theory, and technology. In R.D. Tennyson (Ed.), *Automating instructional design, development, and delivery* (pp. 243–259). Berlin, Germany: Springer-Verlag.

Spector, J.M. (1995). Integrating and humanizing the process of automating instructional design. In R.D. Tennyson & A.E. Barron (Eds.), *Automating instructional design: Computer-based development and delivery tools* (pp. 523–546). Berlin, Germany: Springer-Verlag.

Spector, J.M. (1999). Intelligent support for instructional development: Approaches and limits. In J. Akker, N. Nieveen, & T. Plomp (Eds.), *Design methodology and developmental research in education and training* (pp. 279–290). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Spector, J.M., & Anderson, T.M. (Eds.). (2000). *Integrated and holistic perspectives on learning, instruction and technology: Understanding complexity.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

Spector, J.M., Eseryel, D., & Schuver-van Blanken, M.J. (2001, June). *Current practice in designing training for complex skills: Implications for design and evaluation of Adapt*^{IT}. Paper presentation at ED-Media 2001, Tampere, Finland, June 2001.

Van Merriënboer, J.J.G. (1997). *Training complex cognitive skills: A four-component instructional design model for technical training.* Englewood Cliffs, NJ: Educational Technology Publications.

Vennix, J.A.M. (1996). *Group model building: Facilitating team learning using system dynamics.* Chichester, UK: John Wiley & Sons.

Wiley, D. (2001). *The instructional use of learning objects.* Bloomington, IN: The Association for Educational Communications and Technology (AECT).

Wilson, P. (1991). *Computer supported cooperative work: An introduction.* Oxford, Intellect Books.

Wooley, D.R. (1994, July). PLATO: The emergence of on-line community. *Computer-Mediated Communication Magazine, 1*(3), 5.